Escuela
Politécnica
Superior

Detección y clasificación de desperfectos en la carretera

Grado en Ingeniería Robótica

Trabajo Fin de Grado

Autor:
Fernando Larrosa Marín
Tutor/es:
Miguel Ángel Cazorla Quevedo
Francisco Gómez Donoso

Julio 2021

Universitat d'Alacant
Universidad de Alicante

Detección y clasificación de desperfectos en la carretera

Hacia una ciudad inteligente

Autor
Fernando Larrosa Marín

Tutor/es
Miguel Ángel Cazorla Quevedo
Ciencia de la Computación e Inteligencia Artificial
Francisco Gómez Donoso
Ciencia de la Computación e Inteligencia Artificial

Grado en Ingeniería Robótica

Escuela
Politécnica
Superior

Universitat d'Alacant
Universidad de Alicante

ALICANTE, Julio 2021

Preámbulo

"Este proyecto comienza por el interés que suscitan en mí la visión artificial, el aprendizaje automático, y la forma que tendrá nuestro mundo en un futuro no muy lejano. La combinación de estas tres facetas me llevó a pensar en orientar el estudio hacia la detección de fallos visuales en una ciudad, centrando mis esfuerzos finalmente en la detección y clasificación de desperfectos en la carretera motivado por el desafío *Global Road Damage Detection Challenge 2020* del IEEE."

Agradecimientos

Este trabajo no habría sido posible sin los consejos y los medios que me han proporcionado de principio a fin mis tutores, Miguel y Fran, siempre atentos para solucionar las dudas que me iban surgiendo, y con los que ha sido un lujo trabajar.

También quiero agradecer a los compañeros y amigos que me ha brindado esta carrera, sin los cuales estos cuatro años no habrían sido tan amenos.

Por último, acordarme de mi familia, que siempre han estado ahí apoyándome e interesándose por mi situación en todo momento, animándome cuando los resultados eran malos y celebrándolos cuando han sido buenos.

Muchas gracias.

I learned very early the difference
between knowing the name of something
and knowing something.

Richard P. Feynman.

Índice general

Índice de figuras

Índice de tablas

1. Introducción

Es muy común en las vías urbanas e interurbanas que recorren España encontrar deficiencias entre las que destacan baches, socavones, señales garabateadas o con baja visibilidad, grietas, marcas viales borrosas y bombillas de semáforos y farolas fundidas, como se recoge en el informe de 2018 de la Agencia Española de la Carretera (AEC) [1], que refleja un suspenso de las carreteras españolas en varios de estos aspectos. Acentuándose entre los mismos, el imparable deterioro de los pavimentos, que obtienen una calificación media de "deficiente" como se observa en la Figura 1.1.

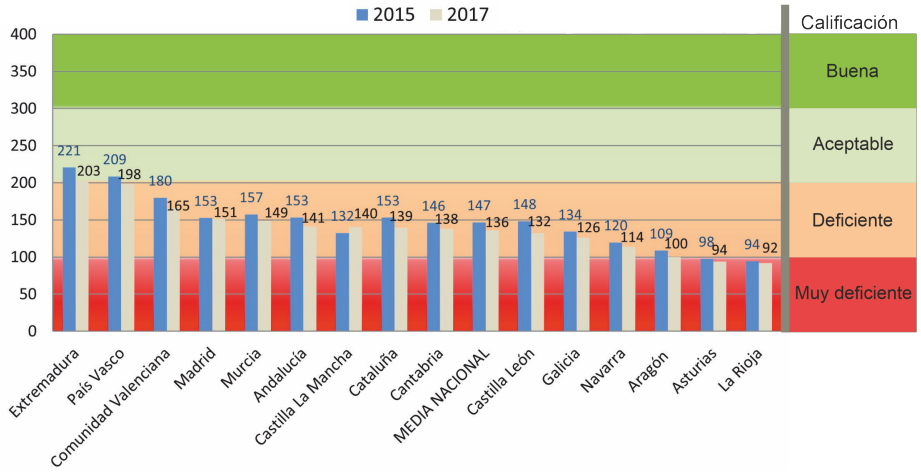


Figura 1.1: Indicador territorial de la evolución del estado de los firmes por comunidades en España. Fuente: [1].

En este proyecto nos centraremos en la búsqueda de desperfectos que se producen en el firme de las carreteras, provocados por el paso de los años, el volumen de tráfico, fuertes lluvias o fallos de impermeabilidad, materiales de poca calidad, o errores de cálculo de cargas y construcción. El conocimiento de su deterioro es fundamental para un mantenimiento eficiente y rentable, con el objetivo de preservar su buena condición y seguridad.

Actualmente, existen tres enfoques para la evaluación del estado del pavimento y la búsqueda de desperfectos en el mismo; manual, semiautomático, y completamente automático. Los enfoques manual y semiautomático se engloban dentro de los métodos tradicionales, y el enfoque automático está en desarrollo actualmente.

- **Enfoque manual.** Los operarios realizan una inspección visual de la superficie de la carretera mientras se desplazan a lo largo de la misma, ya sea caminando o en un vehículo a una velocidad reducida y anotan los fallos que encuentran junto a su localización en la inspección. Es un modo subjetivo que depende de la experiencia del

operario, muy lento y costoso.

- **Enfoque automático.** Para estas evaluaciones comúnmente se usan vehículos equipados con sensores muy caros (cámaras, sensores 3D, sensores inerciales o escáneres láser para obtener los perfiles longitudinal y transversal de la carretera entre otros) que elevan el precio de estos vehículos hasta alcanzar el medio millón de dólares [11]. El precio de operación, aunque más barato que los modelos manuales, se estima entre 30 y 50 dólares por kilómetro [12] para las pocas agencias que han desarrollado un sistema de este tipo. El tratado de datos combina procesamiento de imágenes y reconocimiento de patrones, y se puede hacer en tiempo real o a posteriori, siendo esta última la opción más elegida.
- **Enfoque semiautomático.** Es una mezcla de los anteriores, combina la rápida recolección de datos del enfoque automático usando vehículos con la lenta inspección humana y la subjetividad de los inspectores del manual. Es el más utilizado por las agencias de tráfico actualmente [13] [14].

Como podemos apreciar, ninguna de las opciones anteriores es ideal, con costes tan altos que imposibilitan a los titulares de las carreteras realizar inspecciones con la suficiente regularidad que permita mantener las calzadas en un estado óptimo, dejando de lado aún más a las carreteras menos transitadas.

Es por esto que surge la idea de utilizar dispositivos móviles como *smartphones* que actualmente incorporan cámaras digitales de alta calidad y sensores GPS precisos que posibilitarían realizar una ágil y asequible adquisición de datos de ser montados sobre vehículos como autobuses, camiones de recogida de basura, taxis o coches de policía que diariamente recorren las vías urbanas e interurbanas de una ciudad, así como de pueblos alejados de los núcleos de población.

En este trabajo se aborda la exploración de diversas técnicas que, mediante la implementación de redes neuronales convolucionales, puedan trabajar con estas imágenes recogidas por *smartphones* en tiempo real para detectar en ellas fallos en la carretera y clasificarlos según su tipo que pueda servir para realizar evaluaciones extensas de la red de carreteras a gran velocidad y de forma automática.

1.1. Importancia del estado de las carreteras

El buen estado de la red de carreteras de un país es crucial para el desarrollo del mismo. En el ámbito económico, una empresa industrial preferirá localizarse en una zona con buena comunicación vial que le permita estar conectada con otras empresas y ahorrar en costes logísticos, y en el sector turístico una empresa se verá afectada por unas infraestructuras en mal estado pues causan inseguridad y dificultan la llegada de clientes. En el ámbito social, unas carreteras en buenas condiciones también atraen multitud de beneficios, como el acceso a la educación, acortando tiempos de llegada a los servicios de salud, o en actividades de ocio.

Es por esto que al relacionar los datos por países del informe de 2020 del Índice del Desarrollo Humano (IDH) [15] con el apartado de calidad de las carreteras del informe de competitividad global de 2019 publicado por el Foro Económico Mundial [16], obtenemos la correlación que se observa en la Figura 1.2, que muestra cómo aquellos países con un mayor desarrollo están ligados a un mejor estado en sus carreteras.

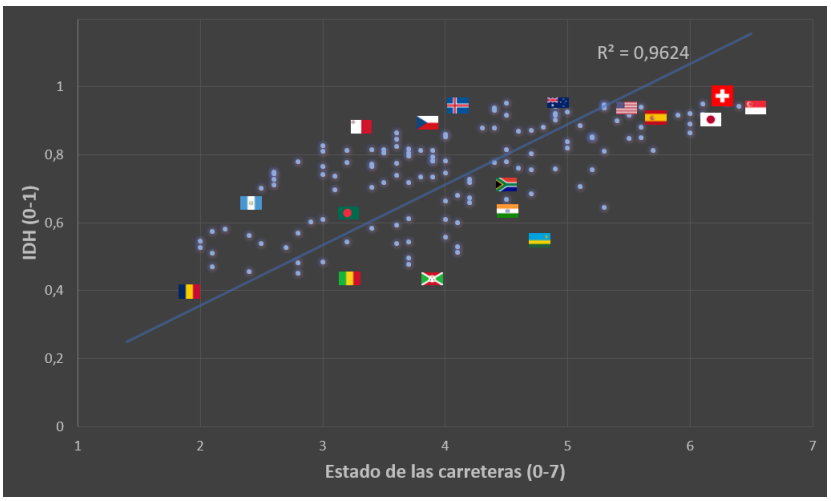


Figura 1.2: Relación entre el IDH y el estado de las carreteras por países

El estado de un firme se puede medir de diversas maneras, pero la más común es mediante el Índice de Regularidad Internacional (IRI), un modelo matemático que se puede resumir como el movimiento vertical acumulado del asiento del conductor de un vehículo producido al recorrer una distancia dada [17]. En [18] se estima que un incremento del IRI de 1 m/km aumenta el consumo de combustible de los coches de pasajeros entre un 2% y un 3%, dependiendo de la velocidad.

En [1] llevaron a cabo una serie de pruebas reales tras las que concluyeron que el aumento del consumo para el mismo incremento en el IRI era algo mayor, del 3,5%, haciendo que los vehículos ligeros produzcan un 4% más de CO_2. Si el pavimento presenta un deterioro estructural importante (grietas, baches y deformaciones), los vehículos ligeros llegan a emitir hasta un 9% más de CO_2. Es tan alto el deterioro de las carreteras españolas que repavimentar la mitad de la red de carreteras del país permitiría ahorrar 1,6 millones de toneladas de CO_2 al año. El hecho de que las carreteras españolas hayan obtenido una calificación de deficiente pero aun así se mantengan entre las mejores carreteras del planeta (ver Figura 1.2) refleja el mal estado del firme en la mayoría de países y la necesidad que existe de un sistema que permita detectar los problemas más graves de forma asequible.

1.2. Global Road Damage Detection Challenge 2020

Cada año desde el 2014 se realiza la Conferencia Internacional del IEEE en Big Data. En estas conferencias se exponen los resultados conseguidos por los ganadores del reto de Big Data que cambia cada año pero que siempre consiste de unas métricas de evaluación claras y un problema bien definido acompañado de un conjunto de datos de partida.

En 2020 se propone como reto el Global Road Damage Detection Challenge (GRDDC) [19], en este, la organización requiere que los participantes propongan un algoritmo capaz de reconocer automáticamente la localización y el tipo de los daños en la carretera presentes en una imagen capturada en uno de estos tres países: India, Japón y República Checa.

En este caso, el conjunto de datos consta de 26336 imágenes de carreteras capturadas en estos tres países. En total, 121 equipos se registraron en esta competición cuyas solu-

ciones se evaluaron usando dos conjuntos de imágenes más pequeños, de en torno a 2600 imágenes cada una. Esta competición concluyó en septiembre del pasado año 2020, pero la página web del reto sigue admitiendo nuevas soluciones, lo que nos permitirá comparar nuestro trabajo con las soluciones del resto de participantes.

2. Marco Teórico

En la última década, se han desarrollado numerosas técnicas para adquirir datos 2D y 3D de las carreteras con el fin de localizar zonas dañadas en ellas [20]. En cuanto a las técnicas 3D, suelen ser utilizados los sensores Kinect de Microsoft [2] y escáneres láser [21]. Estos sensores se suelen montar sobre estructuras que se acoplan a vehículos que cuentan además con otros sensores como acelerómetros o GPS (Figura 2.1). También existen otras aproximaciones que prescinden de los sensores 3D y 2D, como en [22], donde se utiliza el muestreo de frecuencia del acelerómetro de un *smartphone* posado sobre el asiento del copiloto para detectar baches.

Figura 2.1: Vehículo sensorizado del instituto tecnológico de Georgia para la recolección de datos 3D sobre la superficie del pavimento. Fuente: [2].

Por otro lado, se encuentran todas las técnicas que utilizan imágenes o datos 2D para para extraer características del pavimento y detectar a partir de estas los daños existentes en las carreteras. Estas se dividen en dos categorías; las basadas en visión por computador, y las basadas en *machine learning*. En referencia a las basadas en visión, estas típicamente parten de un mapa de disparidad [23] o de una imagen a la que se le aplican varios filtros para reducir el ruido y resaltar la línea de borde de los daños. Después, esta imagen se segmenta usando algún tipo de umbralización como el método de Otsu [3] para extraer aquellas zonas dañadas.

Figura 2.2: Umbralización (a) y detección de daños (b) conseguida en [3].

Atendiendo a las basadas en *Machine Learning* o más concretamente *Deep Learning*, las

Redes Neuronales Convolucionales (CNN) son ampliamente usadas para realizar tareas de áreas tales como reconocimiento de imagen, clasificación o detección de objetos, que resultan difíciles para los enfoques estadísticos tradicionales [24]. Esto es así, porque frente al comportamiento predefinido de los sistemas tradicionales, las CNN están formadas por neuronas artificiales que tienen la capacidad de aprender modificando sus pesos en una fase de entrenamiento. La arquitectura típica de una CNN (Figura 2.3), consiste en una serie de capas convolucionales con funciones de activación intermedias, capas de *pooling* que reducen el tamaño de las grandes matrices que se forman tras las convoluciones, y una última capa de neuronas totalmente conectadas que reciben un vector con los datos aplanados de la última capa de *pooling* y aportan operaciones no-lineales para producir una predicción de salida con ayuda de otra función de activación [4].

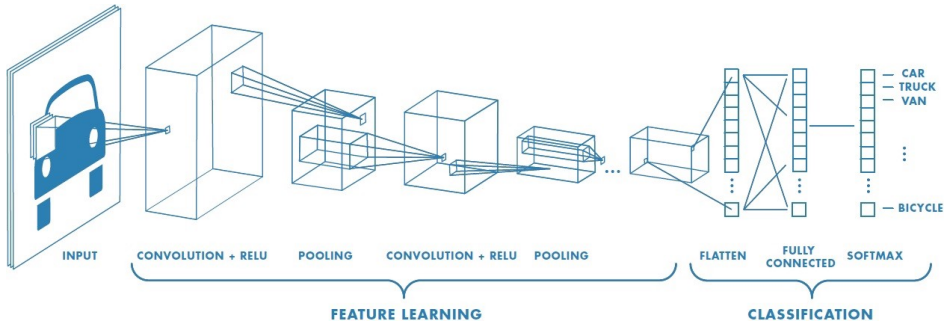


Figura 2.3: Arquitectura típica de una CNN. Fuente: [4]

En [25], se implementa una CNN monoetapa orientada a detección de daños en calzadas basada en la red RetinaNet entrenada con varios *backbones* entre los que se encuentran VGG, ResNet, DenseNet, e InceptionResNetV2, alcanzando muy buenos resultados con VGG pero con una inferencia por imagen de medio segundo. Para conseguir una velocidad de detección más rápida aunque no mucho mayor, en [26] y [27] se explora el uso de Faster R-CNN consiguiendo una puntuación F1 media (Definida en el apartado 4.1.2) en el GRDDC del 51% y 62.5% respectivamente. Para intentar mejorar los resultados obtenidos por [27], [28] utiliza Mask R-CNN pero no lo consigue, ya que obtiene una puntuación F1 del 52.8% y un tiempo de inferencia similar de 0.105 segundos por imagen en una tarjeta gráfica NVIDIA GeForce 1080Ti.

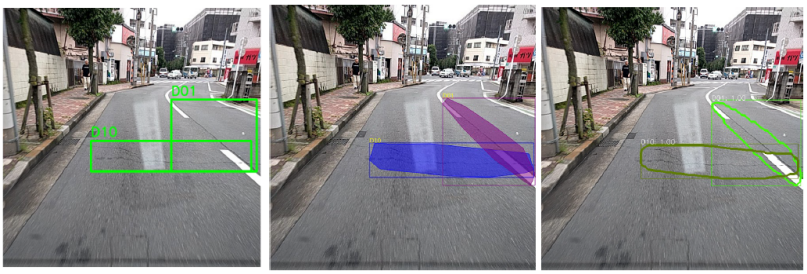


Figura 2.4: De izquierda a derecha; anotación de la imagen original, anotación con máscaras generada con Labelme, resultados de Mask R-CNN. Fuente: [5]

Estos métodos anteriores (Faster R-CNN y Mask R-CNN) son detectores en dos etapas, por lo que pueden conseguir una alta precisión a costa de ser computacionalmente ineficientes. Para alcanzar alta precisión y eficiencia, en [6] se propone una arquitectura multi-nivel y multi-escala que extrae características en pirámide basada en M2det. Con esta arquitectura consiguen unos muy buenos resultados, pero sin alcanzar una velocidad que la haga conveniente para aplicaciones en tiempo real.

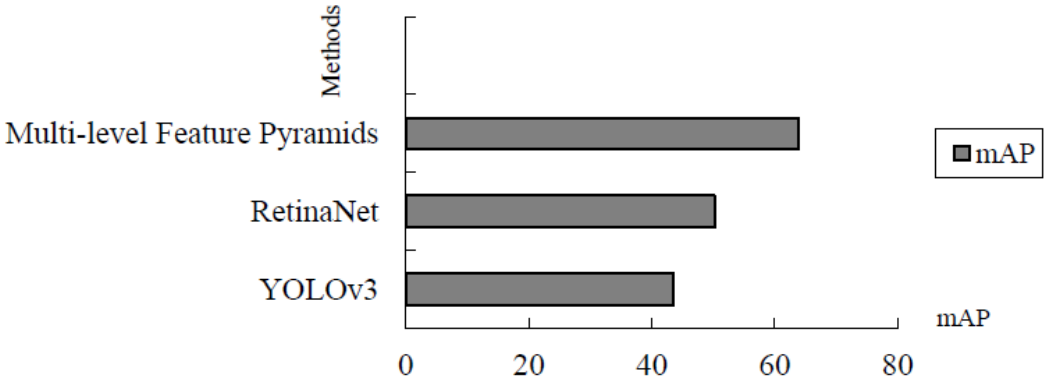


Figura 2.5: Resultados conseguidos en [6] frente a los conseguidos con RetinaNet y YOLOv3.

Para incrementar los *datasets* de daños en carreteras relativamente pequeños con los que se cuentan, surgen variedad de tipos de aumentado artificial de imágenes. En [7], se implementa una "Progressive Growing Generative Adversarial Network" (PG-GAN) que una vez entrenada con una base de datos de daños reales, es capaz de generar imágenes sintéticas de daños realistas que posteriormente se fusionan con imágenes de carreteras sin daños. Se demuestra que este aumento artificial es positivo cuando se cuenta con un *pequeño*, pero que es inútil en grandes y variados conjuntos de datos.

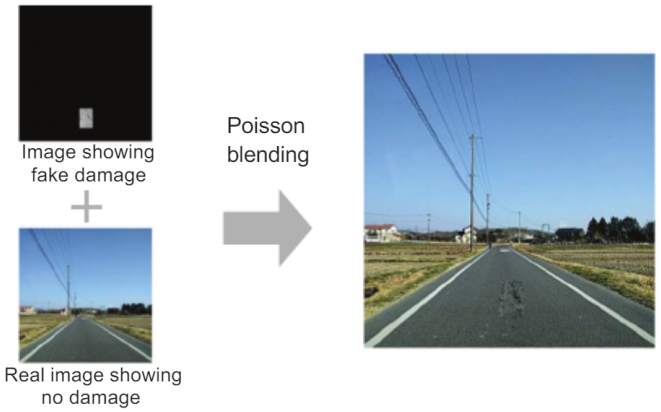


Figura 2.6: Formación artificial de daños en una imagen de carretera. Fuente: [7].

Finalmente, se han explorado técnicas como el "Deep Ensemble Learning" en [29], que

lo utiliza para combinar las prediciiones de varios modelos de YOLOv4 y conseguir unas puntuaciones en los Test 1 y 2 del GRDDC del 62.8% y del 63.6%, respectivamente, sin mencionar nada sobre tiempos de inferencia.

3. Motivación y objetivos

El principal componente de motivación que me llevó a realizar este proyecto ha sido la posibilidad de aprender más de un campo del aprendizaje automático que me apasiona, el *deep learning*. Además, influye mucho la motivación que proporciona siempre una competición, en el contexto de un reto como el *Global Road Damage Detection Challenge 2020* del IEEE en el que participan multitud de equipos alrededor de todo el mundo.

Los objetivos planteados en este trabajo son:

- Proponer un sistema de detección y clasificación de desperfectos en las carreteras económico, robusto y rápido.
- Adquirir habilidad manejando y modificando grandes conjuntos de datos.
- Profundizar en el campo del *deep learning*, aprendiendo a descubrir qué técnicas son útiles y cuáles un inconveniente para un determinado problema.
- Comparar el desempeño en términos de precisión y velocidad de detección de diferentes modelos con diversos tamaños.
- Conseguir una buena posición el ranking de la GRDDC, lo que significaría que nuestro modelo es State Of The Art (SOTA).
- Utilizar el sistema de detección en una aplicación real y observar su comportamiento.

4. Metodología

En esta sección se abordará la estrategia de trabajo adoptada para conseguir los objetivos que se han marcado en este proyecto. Esta estrategia de trabajo viene definida por las herramientas de las que se ha hecho uso y de la forma en que se han utilizado, como se expone a continuación dividiéndolas en dos grupos: aquellas que han servido para la creación y el entrenamiento de la red neuronal, y aquellas que se han empleado para confeccionar la aplicación real.

4.1. Entenamiento de la red

4.1.1. YOLO - Frameworks

YOLO [30] es una abreviatura del término "You Only Look Once" y surgió como un nuevo enfoque para la detección de objetos. Hasta su llegada, los detectores de objetos se limitaban a reutilizar clasificadores ejecutados múltiples veces sobre recortes de la imagen a varias escalas. En cambio, YOLO enmarca la detección de objetos como un problema de regresión a una cuadrícula de celdas espacialmente separadas asociadas a unas probabilidades de pertenencia a cada clase (Figura 4.1). Es por esto que consigue predecir *bounding boxes* y probabilidades de clase en una única evaluación de la red neuronal convolucional. Al evaluar la imagen completamente una única vez tanto en tiempo de entrenamiento como de ejecución, YOLO consigue dos hitos: una mayor velocidad que otros detectores en tiempo real y una gran generalización de lo que representa un objeto perteneciente a una determinada clase.

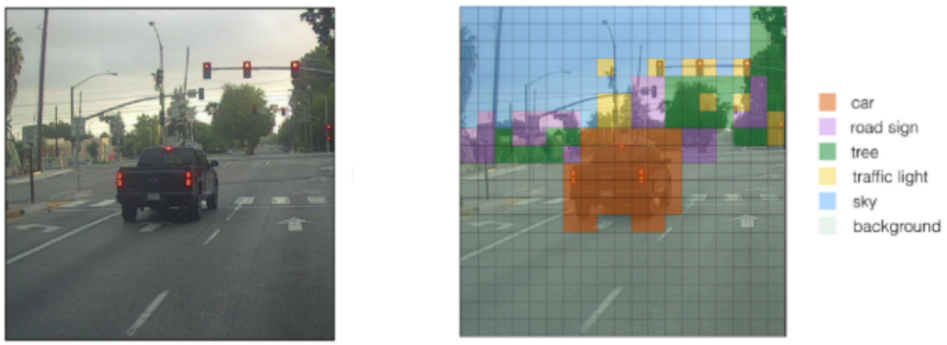


Figura 4.1: Imagen (608x608 píxeles) separada en una cuadrícula de 19x19 celdas de probabilidad de 32x32 píxeles.

Para el presente trabajo, hemos escogido YOLOv5 por ser la versión más reciente de YOLO. Aunque no está tan asentada como su "versión anterior" (realmente son dos versiones desarrolladas por equipos distintos que parten del trabajo de Joseph Redmon et al. en [30])

YOLOv4, establecida sobre el *framework* Darknet, ha demostrado que es un detector SOTA ya que con continuas mejoras ha conseguido puntuaciones muy altas en dos datasets oficiales de detección de objetos: Pascal Visual Object Classes (VOC) y Microsoft Common Objects in Context (COCO).

YOLOv5 - Pytorch

YOLOv5 se ha desarrollado completamente en el *framework* Pytorch (una biblioteca de *machine learning* de código abierto creada por Facebook), y cuya arquitectura de red se muestra en la Figura 4.2.

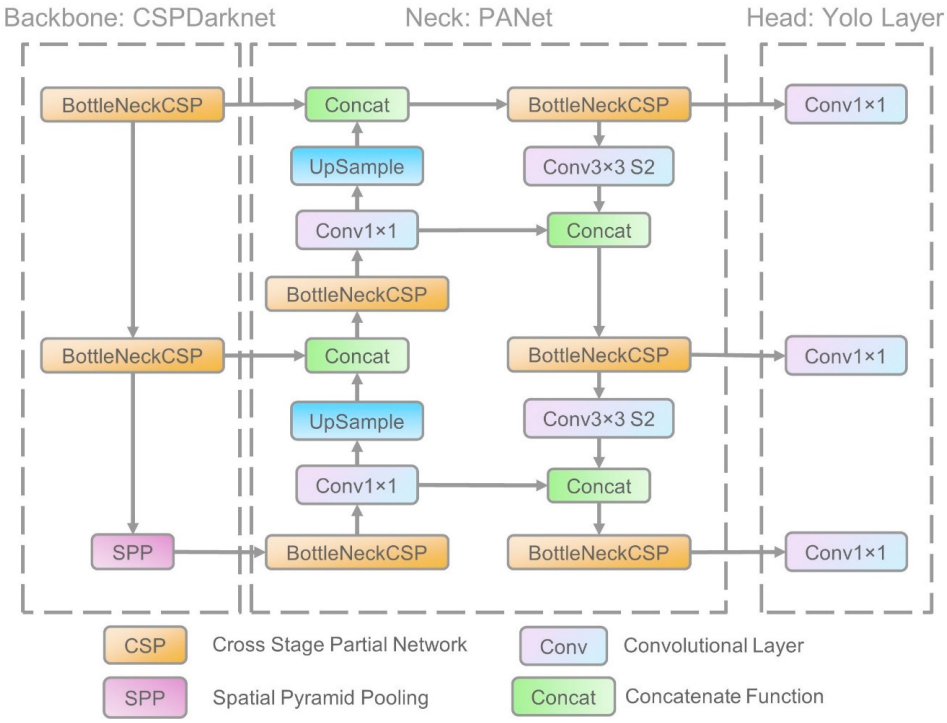


Figura 4.2: Arquitectura de la red de YOLOv5. Fuente: [8].

Esta arquitectura se puede dividir en tres partes bien diferenciadas que comparten todas las versiones de YOLO y en las que YOLOv5 incorpora unos cambios que la hacen idónea para nuestro propósito:

- ***Backbone* (Columna vertebral).** Se trata de una CNN o red neuronal convolucional que extrae y agrega las características de la imagen en diferentes granularidades. YOLOv5 incorpora Cross Stage Partial Network (CSPNet) al *backbone* de Darknet para conseguir CSPDarknet. CSPDarknet soluciona los problemas de repetición de gradientes e incluye los cambios de gradiente en el mapa de características, reduciendo el tamaño del modelo en número de parámetros y los Floating-point Operations Per Second (FLOPS) aumentando la velocidad y precisión de detección. En la detección de daños en la carretera, la velocidad de detección es un factor muy importante para ser

capaces de tratar el flujo de imágenes de entrada en tiempo real.

- ***Neck* (Cuello).** Está compuesto por una serie de capas que mezclan y combinan las características multi-escala obtenidas en la fase anterior y las preparan para la posterior predicción. YOLOv5 utiliza Path Aggregation Network (PANet) para aumentar el flujo de información de la convencional Feature Pyramid Network (FPN) que está limitada por tener una sóla ruta de arriba hacia abajo desde la capa P_7 hasta la P_3 de características (Figura 4.3a). PANet incorpora una ruta de abajo hacia arriba adicional (Figura 4.3b) para solucionar el problema mejorando de esta forma la propagación de características de bajo nivel y por lo tanto, consiguiendo perfeccionar la precisión de localización del modelo. Al mismo tiempo, incorpora Adaptive Feature Pooling (AFP) [31], que es usado para propagar directamente la información útil de cada capa a su correspondiente subred.

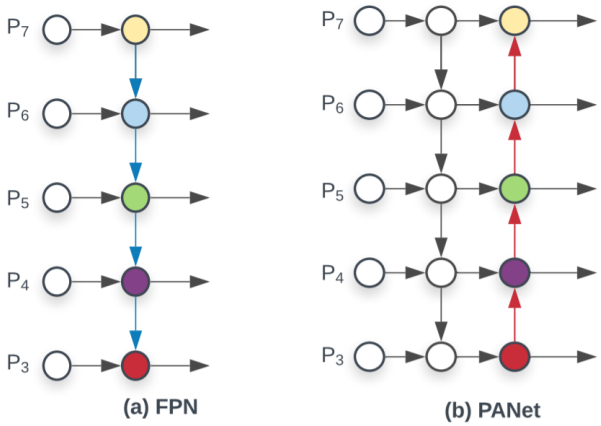


Figura 4.3: Estructura FPN frente a PANet. Cada P_i representa una capa de características de entrada con una resolución de $1/2^i$ de las imágenes de entrada. Para una resolución de entrada de 640x640, P_3 representa características de 80x80 y P_i de 5x5. Fuente: [9]

- ***Head* (Cabeza).** También conocida como capa YOLO, es la última sección de la red, recopila las características preparadas por el cuello de la misma y realiza los pasos de predicción de *bounding boxes* y clases de los objetos detectados. Esta, en YOLOv5 genera tres tamaños diferentes de mapas de características (72x72, 36x36 y 18x18), lo que dota al modelo de una predicción multi-escala haciendo posible la detección de objetos pequeños, medianos y grandes. Esta capacidad de detectar objetos a varias escalas, es idónea para la detección de daños en las carreteras, pues estos se pueden encontrar de varios tamaños, desde pequeños baches a grandes grietas.

4.1.2. Métricas utilizadas

A fin de controlar la calidad de un entrenamiento, es necesario hacer uso de ciertas métricas que reflejen el desempeño del modelo en distintos ámbitos para comprobar si la red está aprendiendo correctamente y comparar posteriormente sus resultados con otros métodos.

- ***Intersection over Union (IoU).*** Cualquier algoritmo de predicción que produzca *bounding boxes* se puede evaluar con esta métrica. Para medir el IoU de una detección se necesitan únicamente dos datos; el *ground truth* (la información conocida) del objeto a predecir en forma de *bounding box* y el propio *bounding box* generado por el pronóstico del modelo entrenado. Para obtener la medida, se divide el área de intersección generada por los dos cuadros delimitadores entre el área de unión de los mismos, y se obtiene un porcentaje de afinidad como se observa en la Figura 4.4.

$$IoU = \frac{|A \cap B|}{|A \cup B|} = \frac{\text{Intersección}}{\text{Unión}} = 43{,}8\%$$

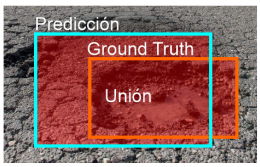


Figura 4.4: Ilustración representativa de la métrica IoU.

  En los sistemas de detección de objetos en imagen, es común utilizar el IoU para clasificar una predicción como correcta si supera un cierto umbral, o incorrecta de lo contrario. En el GRDDC, el umbral está situado en el 50%, por lo que la predicción del ejemplo de la Figura 4.4 sería considerado un error o falso positivo.

- ***True Positives (TP).*** Son los aciertos del modelo, es decir, el número total de objetos que se detectan correctamente con un IoU superior al umbral definido y clasificando acertadamente su clase.

- ***False Positives (FP).*** Son los errores del modelo, es decir, el número de predicciones que se consideran erróneas bien por un fallo en la clasificación o un valor de IoU inferior al umbral definido.

- ***False Negatives (FN).*** Se definen como el número total de instancias de objetos conocidos (*ground truths*) que no han sido detectados y clasificados correctamente.

- ***Precision.*** La precisión es la capacidad de un modelo de detectar únicamente objetos relevantes. Así, es el porcentaje de predicciones correctas (TP) frente a la totalidad de predicciones (TP + FP). Si su valor se acerca a 1, existe una gran probabilidad de que cualquier detección positiva que el modelo produzca sea una detección correcta.

$$Precision = \frac{TP}{TP + FP} = \frac{TP}{Todas\ las\ detecciones}$$

- ***Recall.*** La cobertura es la capacidad del modelo de detectar todos los casos relevantes. De esta forma, se define como el porcentaje de predicciones correctas (TP) frente a la totalidad de objetos reales (TP + FN). Si su valor se acerca a 1, entonces casi todos los objetos presentes en el conjunto de test del *dataset* serán detectados por la red.

$$Recall = \frac{TP}{TP + FN} = \frac{TP}{Todos\ los\ objetos\ reales}$$

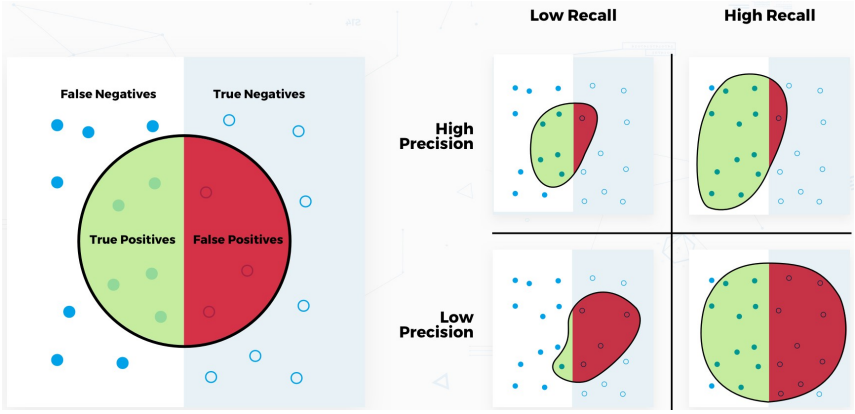


Figura 4.5: Ilustración representativa de los conceptos de *precision* y *recall*. Fuente: [10].

- ***Curva Precision-Recall.*** La curva P-R se produce variando el umbral de confianza (*confidence threshold*) y calculando los valores de *Precision* y *Recall* para cada umbral. Un ejemplo de esta se muestra en la Figura 4.6, donde se observa la curva P-R de cada clase por separado y la conjunción de todas ellas.

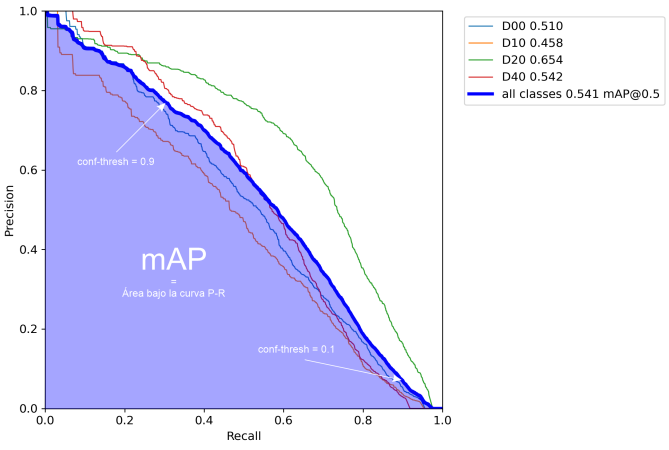


Figura 4.6: Ejemplo de curva P-R.

Normalmente, un aumento de la precisión conlleva una reducción de la cobertura y viceversa, por lo que es importante buscar aquel umbral que mejor compensa las dos

variables para obtener los resultados más óptimos. Un sistema con una alta cobertura pero baja precisión producirá muchos resultados encontrando la mayoría de objetos reales, pero a la vez producirá gran cantidad de predicciones erróneas. Por el contrario, un sistema con alta precisión y baja cobertura producirá pocas detecciones, pero la mayoría de ellas serán correctas.

- ***Mean Average Precision (mAP).*** El mAP es una de las métricas más usadas para calificar el acierto de un modelo predictivo. Aunque su nombre pueda llevar a una confusión, no se calcula como la media de las precisiones de cada clase, sino como la media del área bajo la curva P-R de cada clase como se observa en la Figura 4.6 de arriba. Cuanto más se acerca su resultado a 1, mayor es el área bajo la curva, que se consigue con una gran precisión y cobertura.

- ***F1-Score.*** La puntuación F1 o *Sørensen–Dice coefficient* es una media harmónica de la precisión y el *recall*. **Es la métrica utilizada en el GRDDC,** ya que al tener en cuenta *precision* y *recall* por igual, se favorecen aquellos modelos que maximicen ambas variables simultáneamente y por lo tanto su desempeño sea moderadamente bueno en ambos, en vez de premiar excelentes resultados en una de las dos métricas y un desempeño pobre en la otra.

$$F1 \;=\; 2 * \frac{P * R}{P + R \;+\; 1^{-16}}$$

- **Matriz de confusión.** Otra métrica interesante para observar el comportamiento real de un modelo de forma intuitiva es su matriz de confusión como la del ejemplo de la Figura 4.7. En ella, se representan los objetos predichos por el modelo frente a los objetos que realmente existen en el conjunto de test del *dataset*.

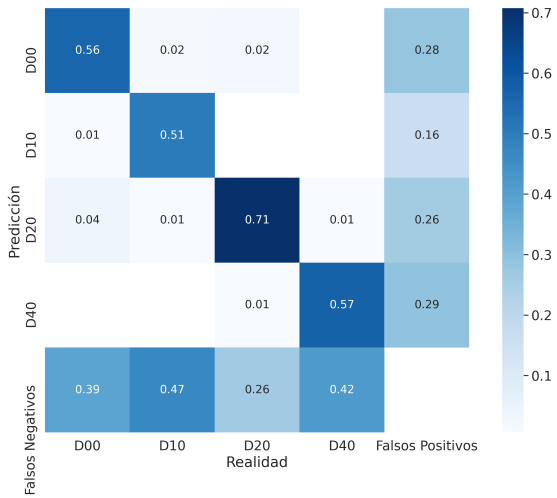


Figura 4.7: Ejemplo de matriz de confusión.

En la diagonal principal se encuentra el porcentaje de aciertos en tanto por uno (ideal-

mente cada casilla de la diagonal principal debería valer 1 y el resto de casillas estar vacías), en la última fila están los falsos negativos de cada clase (aquellos objetos que realmente existen pero no han sido correctamente detectados por el modelo) y en la última columna se observan los falsos positivos (detecciones positivas del modelo que son erróneas por no corresponderse con la realidad).

4.1.3. Weights and Biases - Wandb

Wandb es una herramienta que hace más fácil la organización y el análisis de los experimentos que se llevan a cabo durante una investigación de *machine learning*. Es más ligera que un *TensorBoard toolkit* y gracias a su API para python se puede integrar de forma sencilla con cualquier *framework*, en especial PyTorch, para el que integra características específicas.

Proporciona gran variedad de funcionalidades útiles, entre las que se encuentran las siguientes:

- Guardar los hiperparámetros usados en cada entrenamiento, así como el comando de ejecución inicial, las características del hardware o los parámetros empleados.
- Definir gráficas de todo tipo a partir de las métricas base del entrenamiento o de métricas creadas a partir de las básicas.
- Visualizar las gráficas de los entrenamientos en tiempo real.
- Comparar diferentes entrenamientos entre sí a partir de sus gráficas y parámetros.
- Utilizar un depurador de *bounding boxes* para observar intuitivamente el comportamiento del modelo.
- Mantener indefinidamente el registro de experimentos realizados.

4.1.4. Imgaug

Imgaug es una librería para *image aumentation* o aumento de imágenes de un dataset utilizada en tareas de *machine learning*. Implementa un gran número de técnicas de aumentado que se pueden combinar para convertir un dataset pequeño en uno mucho más grande que contenga nuevas imágenes formadas a partir de ligeras alteraciones de las originales.

Esta herramienta es muy útil porque permite trabajar de una forma sencilla con las modificaciones que se han de aplicar a los bounding boxes de una imagen cuando se aplican sobre ella aumentados geométricos que hacen que los objetos de la imagen cambien de posición, forma o tamaño.

4.2. Aplicación

4.2.1. GeoVideo

GeoVideo es una aplicación móvil para iOS y Android desarrollada por el equipo de proyecto GeoVisuals en la Universidad Estatal de Kent. Su objetivo es cubrir la brecha que existe

entre la diversidad de necesidades de investigadores y los retos de obtención de datos sobre el terreno.

La utilidad que ofrece GeoVideo es la posibilidad de recoger datos de localización GPS mientras se captura información de audio y vídeo al recorrer un camino andando o conduciendo.

4.2.2. Mapbox

Existe una gran cantidad de proveedores de servicios de mapeado disponibles, entre los que destacan Google maps, OpenStreetMaps y la menos conocida Mapbox. Todas ellas varían en características, servicios y tipos de uso. Google maps es la opción más factible para aplicaciones de mapeado que requieran de una mejor precisión y control, pero sus servicios no son convenientes y gratuitos en muchos casos. OpenStreetMaps es un proyecto colaborativo para crear mapas editables y libres que proporciona mapas muy completos pero con una serie de estilos fijos no modificables.

En cambio, Mapbox es una plataforma que proporciona bloques de construcción mediante SDKs y APIs rápidas, estables y escalables para que desarrolladores puedan crear mapas estáticos y en tiempo real gracias a la tecnología OpenGL para sus aplicaciones propias.

Además, viene acompañada de Mapbox Studio, que ofrece un control completo para diseñar y personalizar estilos de mapa propios que posteriormente se pueden utilizar como máscara al hacer una petición del mapa a la API.

4.2.3. Py-Staticmaps

Staticmaps es una librería para python que permite crear imágenes de mapas estáticos con una serie de estilos preconfigurados, además de un estilo vacío. Al contrario que Mapbox, Staticmaps ofrece la posibilidad de dibujar sobre el mapa marcadores a partir de imágenes PNG, polilíneas o círculos geodésicos, todo ello con anti-aliasing (suavizado espacial) para mejorar la calidad de imagen resutante.

5. Desarrollo

En el desarrollo del presente trabajo se ha seguido un procedimiento que no es estrictamente lineal, pues durante la experimentación se ha tenido que revisar en numerosas ocasiones varios aspectos del desarrollo del trabajo para mejorar los resultados obtenidos, pero este procedimiento se puede estructurar en una serie de pasos bien diferenciados que se detallan en los siguientes subapartados.

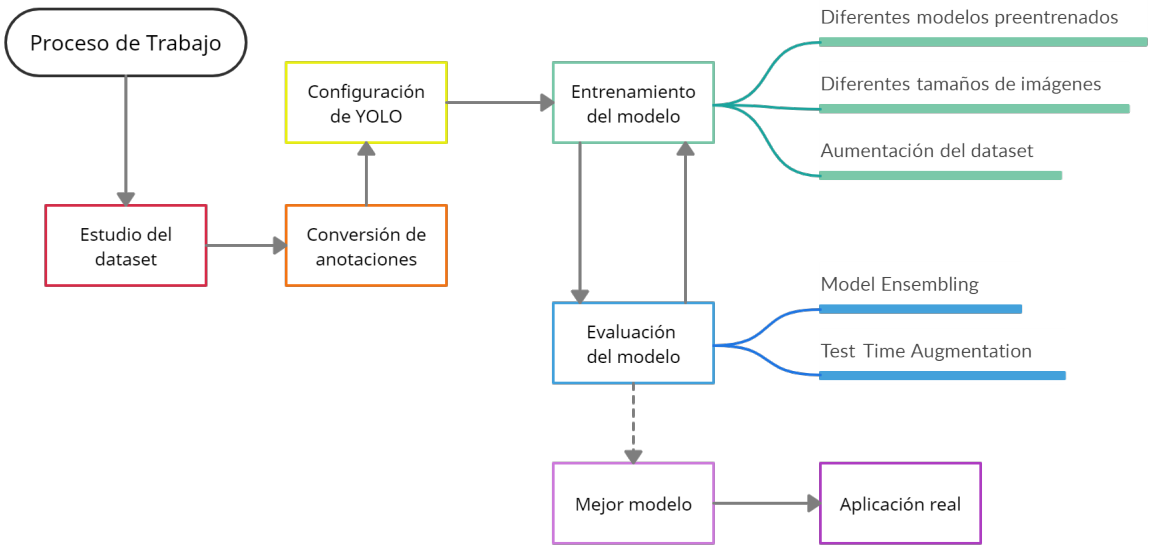
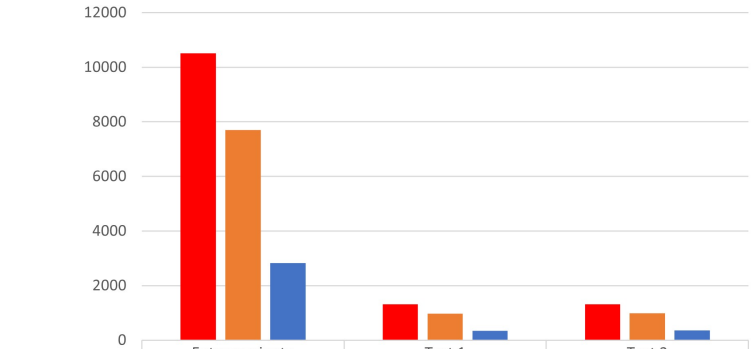


Figura 5.1: Diagrama de flujo del desarrollo del proyecto.

En el diagrama de flujo de la Figura 5.1 se puede observar a grandes rasgos el proceso de trabajo que se ha llevado a cabo. En este, en primer lugar se realiza un estudio del dataset para situar el problema, seguidamente se convierten las anotaciones que acompañan cada imagen al formato requerido y se configura la red YOLO para el entreno. Entonces comienza un procedimiento iterativo, en el que se entrena un modelo con unos parámetros iniciales, se evalúa el modelo de varias formas, y se extraen conclusiones de los errores del modelo para deducir con qué parámetros o técnicas nuevas se debería realizar el siguiente entreno con el objetivo de mejorar los resultados. Una vez obtenido el resultado esperado, se utiliza el mejor modelo para desarrollar una aplicación real de detección de desperfectos en la carretera.

5.1. Estudio del dataset

Para enfocar correctamente el resto del desarrollo del proyecto y ser precisos a la hora de aplicar distintas técnicas a fin de mejorar los resultados de detección y clasificación de daños en la carretera, es conveniente realizar un estudio previo del conjunto de datos del que partimos. Este conjunto de datos o *dataset* es el proporcionado por la organización de la competición, que consta de 26336 imágenes en total repartidas en tres conjuntos; uno de entrenamiento que contiene la mayor parte de las imágenes y dos de test que sirven para probar la capacidad de generalización de nuestro sistema de detección y comparar los resultados con el resto de participantes de la competición.



	Entrenamiento	Test 1	Test 2
Japón	10506	1313	1314
India	7706	969	990
República Checa	2829	349	360

Figura 5.2: Número de imágenes por país en el dataset de entrenamiento y pruebas.

Las imágenes provienen de tres países, Japón, India y República Checa, siendo Japón el país donde se extrajo el mayor número de imágenes y República Checa aquel donde se extrajeron menos fotografías con gran diferencia, como se aprecia en la Figura 5.2 que refleja el número de imágenes por conjunto y país. Estas imágenes fueron capturadas con *smartphones* en sendos países en una combinación de carreteras de distinto tipo, incluyendo carreteras locales y nacionales en zonas urbanas y rurales, bajo diferentes niveles de volumen de tráfico y diversas condiciones climáticas y de iluminación, incluyendo ambientes soleados, nublados, de lluvia ligera o atardeceres.

5.1.1. Clases de daños en la carretera

Las categorías de daño contempladas en [32] son 8, y están basadas en la Guía japonesa de mantenimiento para el pavimento de las carreteras. Esta clasificación que se observa en la Tabla 5.1 considera dos tipos de daños: aquellos relacionados con el deterioro del pavimento (D00, D01, D10, D11, D20 y D40) y aquellos relacionados con el deterioro de las marcas en la carretera (D43 y D44), y todos ellos se encuentran debidamente anotados en formato VOC [33] para cada imagen.

Tipo de daño			Descripcción	Clase
Grieta	Grieta lineal	Longitudinal	Quebradura vertical	D00
			Línea de construcción	D01
		Lateral	Quebradura horizontal	D10
			Línea de construcción	D11
	Grieta de cocodrilo		Grietas entrelazadas	D20
Otros daños			Baches, rodaduras	D40
			Pasos de cebra descoloridos	D43
			Líneas blancas descoloridas	D44

Tabla 5.1: Descripción de las clases del *dataset*.

Pero como el objetivo del reto es conseguir un sistema de detección capaz de ser robusto ante las características de las carreteras de diferentes países y las marcas en la carretera cambian en gran medida según el país, para el Global Road Damage Detection Challenge (GRDDC) 2020 sólo se contemplan las siguientes 4 clases que se muestran en la Figura 5.3:

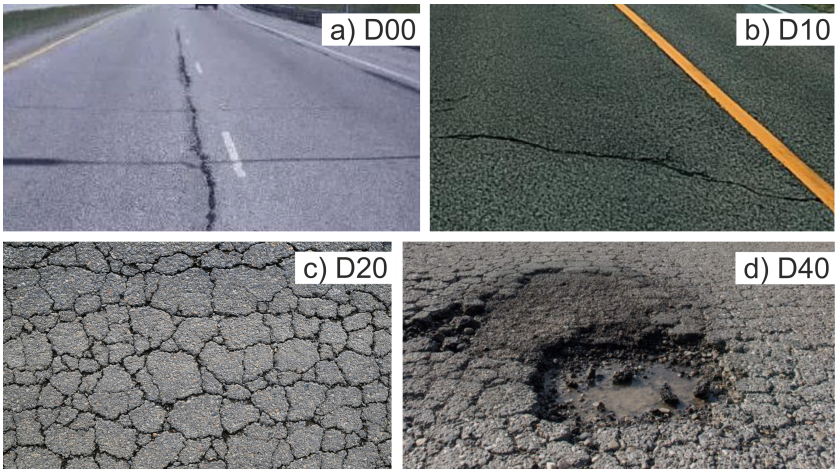


Figura 5.3: Descripción gráfica de las 4 clases del reto.

- **a) D00 - Grietas longitudinales.** Fisuras producidas en la dirección paralela a la trazada de la carretera. Suelen ser producidas por fatigas y pueden indicar el posible comienzo de grietas de cocodrilo.
- **b) D10 - Grietas laterales.** Hendiduras en el firme de una carretera en la dirección perpendicular a la trazada de la carretera. Suelen ser causadas por temperaturas muy bajas.
- **c) D20 - Grietas de cocodrilo.** Conocidas en inglés como "*alligator cracks*", son una forma de grietas interconectadas sobre el pavimento de la carretera. Su nombre proviene de su parecido con el patrón de escamas de un cocodrilo y se producen por múltiples causas.

- **d) D40 - Baches.** Los baches, muy famosos en el lenguaje anglosajón por el nombre de "potholes", se definen como un desnivel en el suelo o pavimento, formando hundimientos de la capa superficial con forma de cuenco. Generalmente, cuentan con bordes afilados y caras verticales cerca de la parte superior que pueden causar daños serios en los vehículos que los sobrepasan a altas velocidades. Suelen ser una evolución de las grietas de cocodrilo que forman pequeños trozos de pavimento que se acaban despegando del firme.
- **e) D40 - Rodaduras.** Las rodaduras se incluyen en la misma clase que los baches, por ser depresiones en la superficie de la carretera. En este caso, las rodaduras son formadas por el continuo paso de las ruedas de los vehículos y se da en aquellos asfaltos mal construidos.

Figura 5.4: Ejemplo de rodadura extrema.

5.1.2. Balanceo de clases

Un aspecto muy importante en un conjunto de datos a la hora de utilizarlo para el entrenamiento de una red neuronal, viene dado por el correcto balanceo del número de elementos por clase del *dataset*. Si existen clases cuantiosamente sobrerrepresentadas en comparación con otras, el modelo tiende a detectar y clasificar más veces estas clases de las que debería frente a aquellas clases con menor representación, a menudo conllevando un aumento del error.

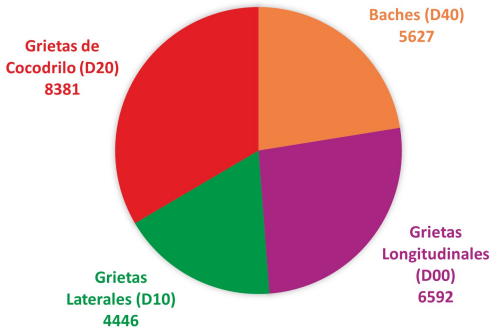


Figura 5.5: Número de daños por tipo en el conjunto de entrenamiento.

Como se observa en la Figura 5.5, el conjunto de datos de entrenamiento no está muy

desbalanceado. Realmente existe un buen equilibrio en el mismo ya que hay una gran cantidad de imágenes de cada categoría, siendo la clase D20 la más numerosa con casi el doble de elementos que la clase D10, lo que no supone un desbalanceo exagerado y no será necesario producir un balanceo artificial del *dataset*.

5.1.3. Tamaño y forma de los objetos

La correcta adaptación de un modelo predictivo a los tamaños y los factores de forma típicos de los distintos objetos del *dataset* es fundamental para obtener resultados precisos con un alto IoU. En el conjunto de imágenes aleatoriamente extraídas del *dataset* de la Figura 5.6, podemos observar algunos ejemplos de los daños en la carretera que se busca detectar.

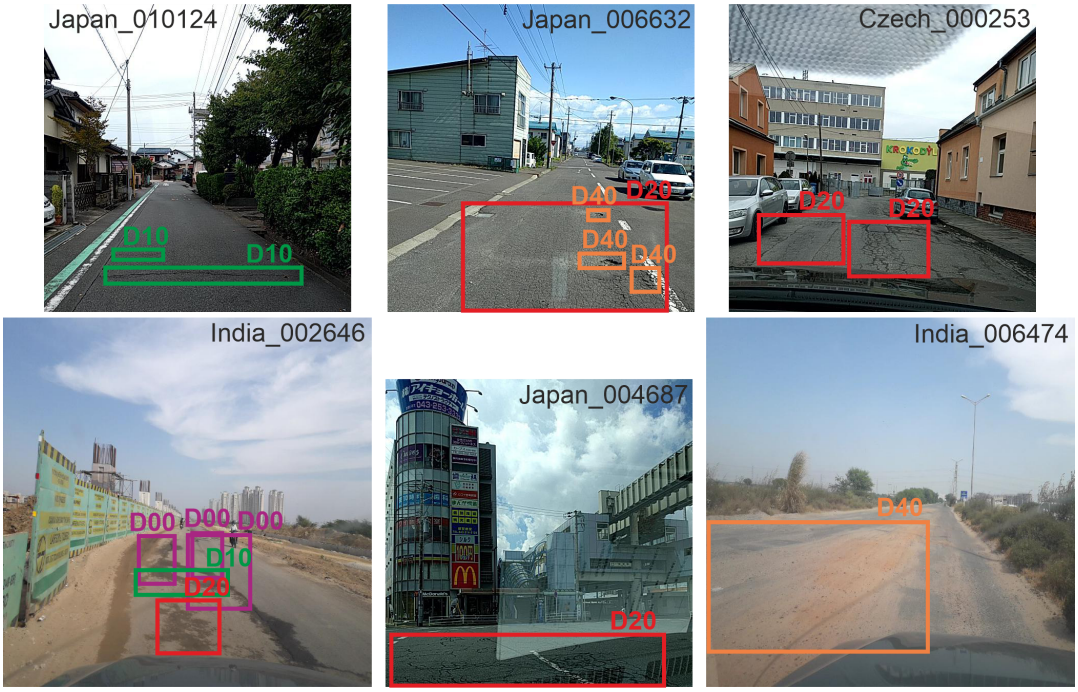
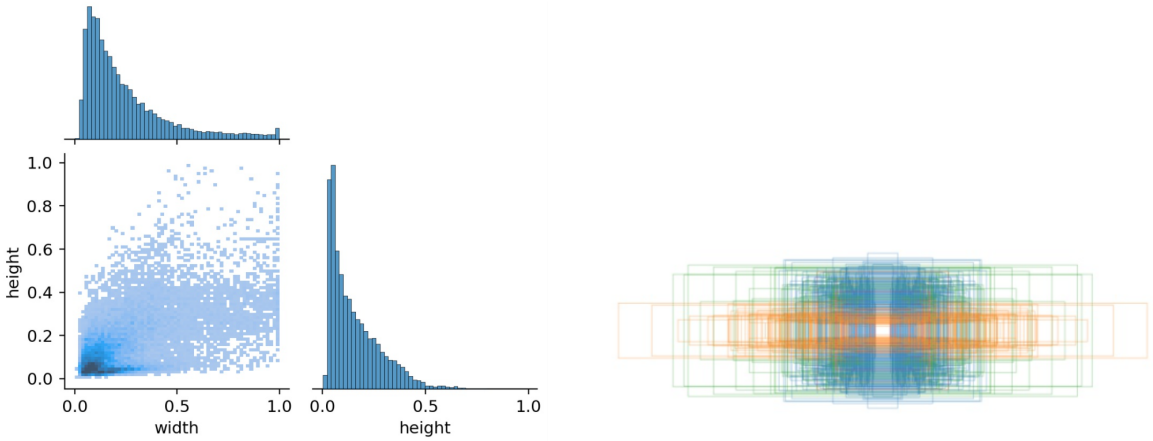


Figura 5.6: Grupo de imágenes del *dataset* elegidas aleatoriamente con sus etiquetas.

En la Figura 5.7a vemos cómo la distribución refleja que la gran mayoría de objetos en el *dataset* no superan en tamaño un cuarto de la imagen a la que pertenecen, teniendo la mayoría de estos una altura y anchura menor que 100 píxeles. La Figura 5.7b representa un grupo de imágenes según su *bounding box* normalizado, y se puede observar en ella la forma típica de cada clase.

Tras estudiar el conjunto de datos en profundidad se obtienen las siguientes conclusiones de cada clase:

- **D00 - Grietas longitudinales.** Suelen ser rectangulares, con su lado mayor orientado en la dirección vertical de la imagen, aunque su forma puede variar.
- **D10 - Grietas laterales.** Tienen una forma alargada y fina muy característica que se repite en la práctica totalidad de los daños de esta clase.



(a) Distribución de objetos en el *dataset* por altura y anchura.

(b) Forma normalizada de un grupo de imágenes del dataset.

Figura 5.7: Distribución de objetos en el *dataset* por altura, anchura y forma normalizada.

- **D20 - Grietas de cocodrilo.** Son de forma rectangular sin un factor de forma bien definido. El tamaño de las grietas de cocodrilo es muy cambiante, aunque los objetos de mayor tamaño del dataset pertenecen a esta clase.
- **D40 - Baches y rodaduras.** No tienen una forma determinada, en la Figura 5.6 se puede ver como hay baches muy pequeños, medianos y muy grandes. Son difíciles de modelar pues instancias de la misma clase pueden tener características muy diferentes.

5.2. Conversión de anotaciones - De VOC a YOLO

Para cada imagen en el *dataset*, existe un archivo de anotación que contiene la localización y la clase de los objetos que se reconocen en la imagen. En el caso del conjunto de datos del GRDDC, estas anotaciones vienen escritas en formato Pascal Visual Object Classes (VOC) famoso por ser el que usa el proyecto ImageNet.

El formato Pascal VOC no es más que un archivo XML con una estructura determinada, incluyendo la ruta hacia la imagen respectiva a la anotación, su tamaño y una lista de objetos contenidos en la misma. Cada uno de estos objetos a su vez contienen el nombre de la clase a la que pertenecen y las coordenadas de las dos esquinas principales del "*bounding box*" o cuadro delimitador definidas en píxeles desde el origen de la imagen, situado en la esquina superior izquierda de la misma. En la siguiente Figura 5.8 se observa la anotación en VOC para una imagen del *dataset*:

```
<annotation>
  <folder>images</folder>
  <filename>Czech_000037.jpg</filename>
  <size>
    <depth>3</depth>
    <width>600</width>
    <height>600</height>
  </size>
  <object>
    <name>D40</name>
    <bndbox>
      <xmin>381</xmin>
      <ymin>451</ymin>
      <xmax>429</xmax>
      <ymax>466</ymax>
    </bndbox>
  </object>
  <object>
    <name>D10</name>
    <bndbox>
      <xmin>298</xmin>
      <ymin>403</ymin>
      <xmax>411</xmax>
      <ymax>427</ymax>
    </bndbox>
  </object>
</annotation>
```

Figura 5.8: Ejemplo de anotación de imagen en formato Pascal VOC.

Para entrenar una red neuronal en YOLO, es necesario disponer de las anotaciones en formato YOLO, un tipo de formato escrito sobre un archivo .txt cuyo nombre ha de ser el mismo que el de la imagen a la que hace referencia y en su interior ha de tener una línea por cada objeto en la imagen que se construye de la siguiente forma:

- **Primer parámetro.** El número de la clase a la que pertenece el objeto.
- **Segundo parámetro.** La coordenada X del centro del objeto normalizada según la anchura de la imagen.
- **Tercer parámetro.** La coordenada Y del centro del objeto normalizada según la altura de la imagen.
- **Cuarto parámetro.** La anchura del *bounding box* normalizada según la anchura de la imagen.
- **Quinto parámetro.** La altura del *bounding box* normalizada según la anchura de la imagen.

Así, el ejemplo de anotación de la Figura 5.8 se convierte en las siguientes dos líneas de la Figura 5.6 en formato YOLO según los parámetros comentados.

```
3 0.675 0.764 0.08 0.025
1 0.591 0.692 0.188 0.04
```

Figura 5.9: Ejemplo de anotación de imagen en formato YOLO.

En la siguiente Figura 5.10 se observa gráficamente la diferencia a la hora de definir el *bounding box* en el formato VOC y en el formato YOLO, en este último incluyendo las normalizaciones en los parámetros referentes a las coordenadas del centro del cuadro delimitador de objeto y las normalizaciones en la anchura y altura del cuadro.

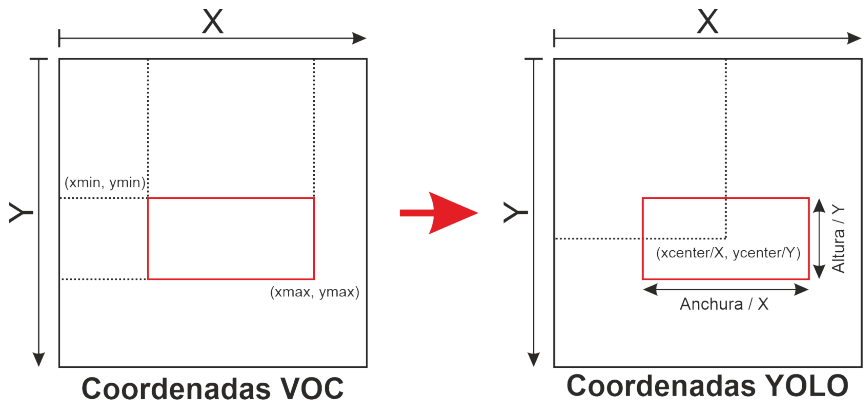


Figura 5.10: Conversión de coordenadas de VOC a YOLO.

5.3. Entrenamiento del modelo

En esta sección estudiaremos los métodos más importantes que hemos empleado para mejorar los resultados de experimentación con la red neuronal en términos del entrenamiento de la misma.

5.3.1. Distintos modelos preentrenados

En el presente trabajo se ha escogido la versión más reciente de YOLO, la familia de redes YOLOv5 como la principal vía para obtener un modelo preciso y veloz en la detección que permita realizar una aplicación en tiempo real o lo más cerca posible del mismo. Como línea base se ha utilizado YOLOv4, que sirve para realizar comparaciones que nos indiquen si los modelos entrenados se están comportando bien.

En la Figura 5.11, se observa cómo se comporta la familia de redes YOLOv5 en confrontación con la familia de redes SOTA de EfficientNet [34] en términos de precisión y velocidad en la prueba del completo *dataset* COCO [35]. YOLOv5 supera a todos sus predecesores en el *dataset* COCO, y se acerca a los resultados de detección de EfficientNet con velocidades de detección por imagen mucho más rápidas, lo que decidió su elección.

En la Figura 5.12, se muestran las especificaciones técnicas de los cuatro modelos preentrenados que conforman la familia de YOLOv5. Se ordenan de izquierda a derecha de menor a mayor tamaño, siendo la versión YOLOv5s la más liviana por su reducido número de parámetros que hacen que a su vez sea la más rápida y la que peor puntuación obtiene en el reto COCO y YOLOv5x la más pesada, con el mayor tiempo de detección por imagen y la mayor puntuación en el mismo reto.

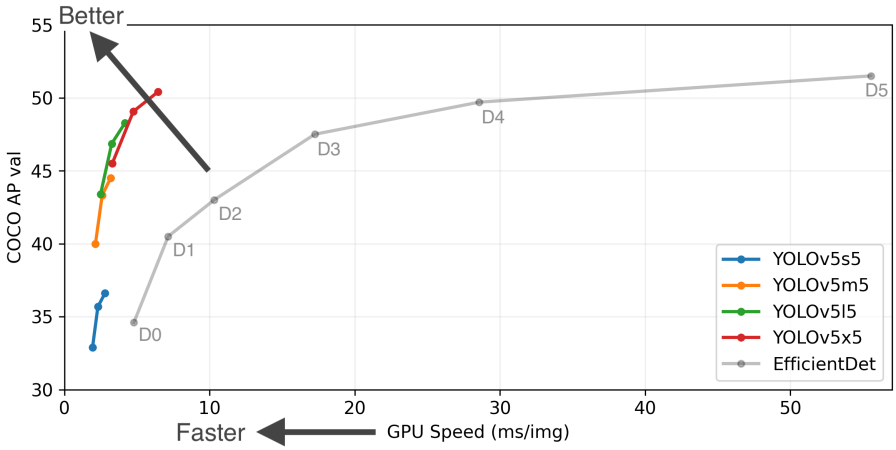


Figura 5.11: Comparación de velocidad de detección y mAP. Velocidad GPU media medida usando una GPU V100 y un tamaño de *batch* de 32 en YOLOv5. Velocidad GPU obtenida de google con un tamaño de *batch* de 8 en EfficientNet.

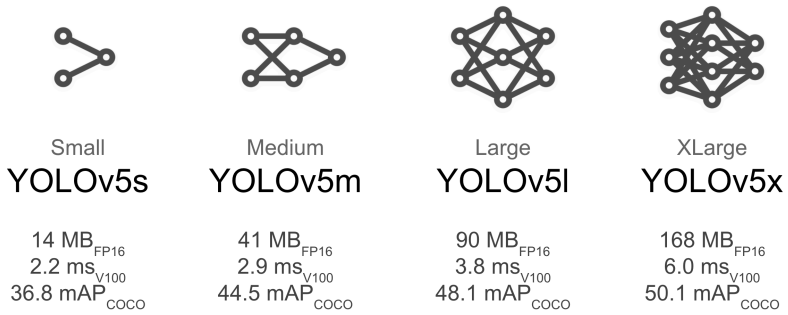


Figura 5.12: Modelos de pesos preentrenados de YOLOv5.

5.3.2. Diferentes dimensiones de imágenes

Como se ha observado en el apartado anterior de estudio del *dataset*, este cuenta con imágenes de distintas dimensiones. En concreto, las imágenes provenientes de Japón y República Checa tienen unas dimensiones de 600 x 600 píxeles y aquellas tomadas en la India cuentan con unas dimensiones mayores, de 720 x 720 píxeles como se observa en la Figura 5.13.

A la hora de entrenar la red convolucional, es necesario que algunas o todas las imágenes se reescalen para que la totalidad de ellas compartan las mismas dimensiones. Puesto que YOLO en su proceso de convolución reduce las imágenes entre 32, las imágenes se deberán reescalar para que las dimensiones en píxeles de las mismas sean múltiplos de 32, algo sencillo ya que todas las imágenes del dataset son cuadradas, por lo que no será necesario realizar recortes sobre las imágenes ni alterar su relación de aspecto.

Tener un conjunto de entrenamiento de imágenes tan grande que eventualmente puede crecer aún más por el aumento del *dataset*, es muy bueno por la generalización que podemos conseguir, pero hay que tener en cuenta que la memoria VRAM de las GPUs en las que se entrena la red es limitada, y dependiendo del número de parámetros que tenga el modelo que estemos utilizando para entrenar es posible que debamos reducir el tamaño de las imágenes

para poder llevar el entreno a cabo. Es por esto que en la fase de experimentación se probarán distintos tamaños de imagen, desde 416 hasta 640 píxeles cuadrados.

Otro aspecto a tener muy en cuenta es el tamaño del *batch* del entreno, que no es más que el número de imágenes que contiene el lote en el que dividimos el dataset para entrenar. Cuanto mayor sea este lote, mayor generalización y por tanto mejores resultados obtendremos, pero mayor será también la necesidad de memoria. Es por todo esto que el tamaño de las imágenes se irá variando en los diferentes entrenamientos dependiendo del aumento del *dataset*, el tamaño del *batch* o el modelo utilizado.

Figura 5.13: Dimensiones de las imágenes en el *dataset* por país.

5.3.3. Aumento del *dataset*

El *Data Augmentation* (o aumento de datos) es una técnica de sintetizado de datos artificiales a partir de datos disponibles usado para incrementar el tamaño de un *dataset*. En aplicaciones de visión con *deep learning*, el proceso de aumento es imprescindible, pues de no usarlo, no se estaría aprovechando el *dataset* al máximo. El procedimiento es sencillo, se toma una imagen del conjunto de entrenamiento, se modifica de algún modo y se añade de vuelta al conjunto de entrenamiento con lo que este crece en una unidad.

Esto se puede hacer con todas las imágenes del *dataset*, aplicando varias modificaciones distintas a cada imagen para multiplicar el tamaño del mismo. Además, se puede realizar el aumento en línea con el entreno, de tal forma que se consigue un *dataset* de tamaño infinito que hace que el modelo nunca vea dos imágenes idénticas durante el entrenamiento. Con esta técnica, se previene el *overfitting* (sobreentreno que empobrece la generalización del modelo) y hace que el modelo se comporte mejor ante imágenes nuevas. Todo esto sin necesidad de recoger y etiquetar nuevas imágenes, tarea que muchas veces es costosa o inviable.

En este trabajo, se probarán varias combinaciones diferentes de las siguientes técnicas de *image augmentation*, que se contrastarán con la ausencia de aumento, para observar el impacto que produce cada combinación en los resultados:

- **Cambio de color HSV** El modelo de color HSV (*Hue, Saturation, Value*) en contra del ampliamente utilizado RGB, permite separar la luma (información de luminosidad) de una imagen de su crominancia (información de color). Existen otros modelos de color que realizan esta separación, pero en tareas de visión se suele utilizar HSV por su

sencillez de conversión con RGB. En la Figura 5.14 se observa cómo afecta la variación de los tres canales HSV. En este trabajo, se experimenta un aumento en tiempo de entrenamiento que modifica aleatoriamente cada canal en un rango, siendo este rango muy pequeño para el matiz (±1%) y más grande para la saturación y el valor (±60% y ±40% respectivamente). De esta forma, se consigue un modelo robusto ante cambios de luz y sombras, características muy comunes en el entorno vial de aplicación.

Figura 5.14: Variación de los tres canales HSV de una imagen original aisladamente.

- **Mosaico.** La técnica de aumento por mosaicos se introdujo con la aparición de YOLOv5 de parte de su creador Glenn Jocher y posteriormente pasó a ser usada por YOLOv4. Consiste en combinar cuatro imágenes del *dataset* en una única mediante recortes de las mismas (Figura 5.15. Con este método, se consigue que el modelo tenga un mejor desempeño ante objetos que se ven parcialmente (oclusión) o no se encuentran en las áreas de la imagen donde se suelen concentrar (traslación), además varía el número de objetos por imagen haciendo al modelo más versátil en situaciones con pocos o muchos objetos.

Figura 5.15: Ejemplo de tres imágenes construidas con la técnica del mosaico.

- **Escala y volteo.** Estas técnicas de aumento son unas de las más utilizadas en el campo

de la visión por computador por su capacidad de aumentar el *dataset* de forma sencilla y rápida. El procedimiento de escalado sirve para hacer al modelo capaz de detectar objetos de mayor o menor tamaño que los de su misma clase presentes en el *dataset*. El volteo o modo espejo, es idóneo para crear nuevos objetos pues al reflejarlas, sus características cambian mucho a ojos de la red. Para la aplicación de este trabajo, se escalan las imágenes en un rango de un $\pm 50\%$ de su tamaño original y se realiza un volteo horizontal de las mismas.

5.4. Evaluación del modelo

Los resultados que arroja una red no sólo se pueden mejorar mediante cambios en la forma en que la entrenamos, también existen algunos métodos que pueden hacer que los resultados de inferencia mejoren mediante cambios en la forma de utilizar el modelo una vez entrenado. En este apartado veremos las dos técnicas que hemos puesto en práctica, el *Test Time Augmentation* y el *Model Ensembling*.

5.4.1. Test Time Augmentation (TTA)

De forma similar a el aumento del *dataset* que se realiza para el entrenamiento, el propósito del *Test Time Augmentation* es llevar a cabo modificaciones sobre las imágenes que se quiere predecir. Por lo tanto, en lugar de mostrar las imágenes en crudo una única vez al modelo entrenado para realizar la detección y clasificación, se le mostrará una colección de imágenes producidas por el aumento de la imagen de test en crudo. Entonces, se elaborará la predicción final promediando la predicción de cada una de las imágenes de la colección.

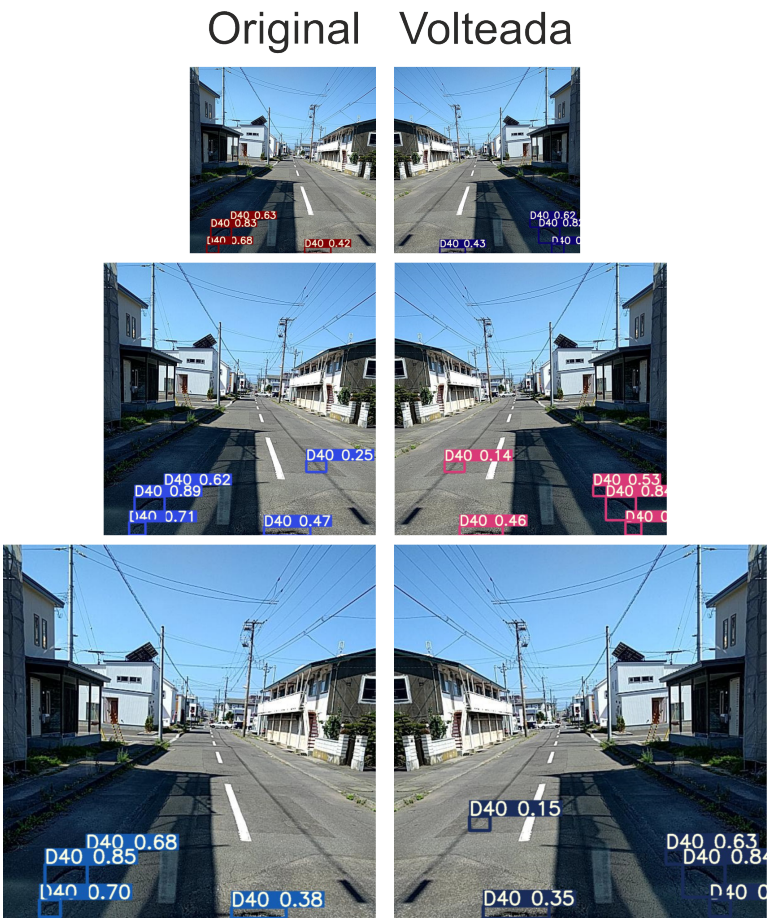


Figura 5.16: Colección de imágenes de TTA para una imagen del *dataset* con las predicciones que producen en ellas el mismo modelo entrenado.

El número y tipo de aumentos por imagen para efectuar un TTA no está definido, en el desarrollo de nuestro trabajo hemos optado por alimentar al modelo con la imagen original y la misma volteada, en tres resoluciones diferentes (la original, una más pequeña que esta, y otra más grande). En la Figura 5.16, se puede observar cómo los resultados de inferencia de cada imagen de la colección son diferentes al ser predichos por uno de los modelos finales de este trabajo. La confianza de los objetos detectados varía en cada imagen, llegando a haber algunos objetos que son detectados en unas imágenes y en otras no, por no superar el umbral mínimo de confianza establecido en 0.1 para este ejemplo.

Si miramos al *ground truth* de la imagen en la Figura 5.17, los objetos etiquetados en el *dataset* que deberíamos predecir correctamente, vemos como en una predicción correcta debería haber cinco objetos de la clase D40 (baches) detectados que coincidieran parcialmente en posición y tamaño con los *bounding boxes* etiquetados.

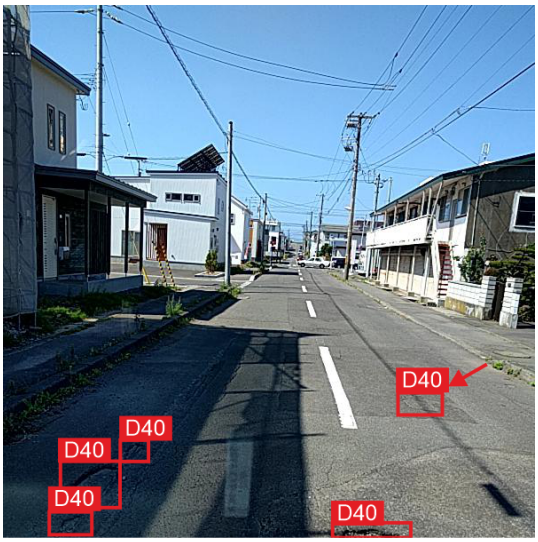


Figura 5.17: *Ground truth* de la imagen de ejemplo.

Pero esta predicción correcta no se produce en todas las imágenes del conjunto de TTA, existiendo la mayor controversia en la detección del bache señalado con una flecha en la Figura 5.17. Aquí es donde el TTA entra en juego, fusionando los resultados de aplicar el modelo entrenado a la colección de seis imágenes generada antes de aplicar Non Maxima Supression (NMS) y aplicando NMS después.

Finalmente, en la Figura 5.18 de arriba, vemos la predicción final aplicando TTA para la imagen de ejemplo. En esta, se puede observar cómo la confianza de cada objeto es mayor que la confianza devuelta por cada una de las seis imágenes de la colección por separado, coincidiendo además todas ellas con el *ground truth*. La razón por la que el TTA suele funcionar se debe a que en el proceso de promediar las predicciones, también estamos promediando los errores, y el error puede ser grande para una predicción en un modelo bien entrenado, pero cuando se promedia este error entre varias predicciones, este tiende a disiparse y dejar ver la respuesta correcta.

El *Test Time Augmentation* es muy útil para predecir imágenes en las que el modelo está inseguro en su inferencia, es decir, tiene poca confianza en sus predicciones. Es por esto que

Figura 5.18: Predicción final aplicando TTA para la imagen de ejemplo.

le mostramos un conjunto de imágenes que aunque a nuestros ojos sean muy similares, para la red convolucional son muy diferentes. En el apartado de experimentación, se comprobará si el TTA es realmente beneficioso para mejorar las predicciones de la red, y se cuantificará el incremento del tiempo de inferencia que inevitablemente se producirá al aumentar el número de imágenes a predecir.

5.4.2. Model Ensembling

EL *Model Ensembling* o ensamblado de modelos es una técnica que combina varios modelos en la inferencia de unos mismos datos de entrada. Aplicado a la detección y clasificación de objetos en imágenes 2D, cada modelo se usa para predecir la probabilidad de que haya un objeto de una determinada clase en un lugar determinado de una imagen así como la forma y tamaño de su *bounding box*. Después, los *bounding boxes* de cada modelo se combinan para obtener la inferencia final.

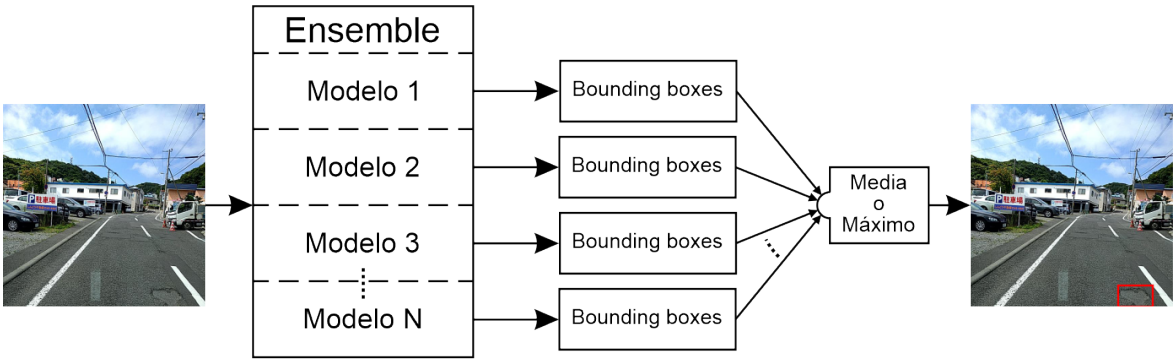


Figura 5.19: Esquema del método de ensamblado de modelos llevado a cabo.

Como se observa en la Figura 5.19, la forma de combinar los *bounding boxes* puede variar dependiendo de lo que queramos conseguir. En nuestro caso se experimentará con dos de ellas; la media y el máximo. El método de promediar las predicciones tiene el mismo fondo que el TTA, es decir, promediar los errores entre distintos modelos para corregirlos. En cambio, el método de supresión de no máximos (NMS), busca quedarse con la predicción que resulte en una mayor confianza de entre todos los modelos, conllevando un mayor número de detecciones positivas por imagen.

Por norma general, el ensamblado de modelos suele producir un mejor desempeño en la práctica por su naturaleza. A mayor número de modelos en el ensamblado, el rendimiento frecuentemente será mayor, hasta llegar a un umbral en el que la mejoría se suele estancar y en algunos casos puede llegar a ser contraproducente.

En el *Model Ensembling*, de manera similar al TTA, el aumento del rendimiento viene irremediablemente acompañado de un mayor tiempo de inferencia, es decir, una disminución de los Fotogramas Por Segundo (FPS) que el modelo puede manejar en la detección. En el apartado de experimentación, mediremos el impacto en términos temporales que se produce, así como la variación del rendimiento para ambas combinaciones (media y máximo), y para ensamblados con mayor o menor número de modelos.

5.5. Aplicación real de detección de daños en carreteras

En este apartado se expondrán los principales pasos que se han seguido para diseñar una aplicación real que use el mejor modelo de YOLOv5 con el objetivo de detectar los desperfectos en la carretera capturados por la cámara de un teléfono móvil. Además, la aplicación utilizará los datos GPS del teléfono móvil para registrar los daños encontrados en un mapa.

5.5.1. Adquisición de datos

La adquisición de datos, tanto de localización GPS como de imagen, se realiza con un *smartphone* relativamente actual, en concreto un iPhone 8. Se tomarán imágenes de vídeo con el teléfono móvil en calidad 1080p Full HD a 30 FPS. El teléfono móvil estará montado directamente sobre la luna de un coche, utilizando la estabilización óptica de imagen para vídeo que integra, evitando así montajes más costosos e inaccesibles que impliquen un estabilizador de cámaras externo.

Para obtener la información conjunta de localización y vídeo se utiliza la aplicación móvil GeoVideo. Con ella, somos capaces de capturar imágenes de vídeo a 30 FPS a la vez que obtenemos la localización GPS actual del *smartphone* cada segundo que transcurre. La información de localización se almacena en una tabla en formato acrshortcsv como la que se muestra en la Tabla 5.2, que contiene la fecha y hora así como la latitud y longitud de la posición en cada punto de la ruta.

index	date	time	latitude	longitude	mediatime
0	06/06/2021	7:48:21	38.087073	-0.972933	0:00:01
1	06/06/2021	7:48:22	38.087106	-0.972968	0:00:02
2	06/06/2021	7:48:23	38.087091	-0.972949	0:00:03
3	06/06/2021	7:48:24	38.087048	-0.972941	0:00:04
4	06/06/2021	7:48:25	38.087042	-0.972940	0:00:05
..	...	...	...	...	...
453	06/06/2021	7:55:54	38.059873	-0.943694	0:07:34
454	06/06/2021	7:55:55	38.059792	-0.943594	0:07:35
455	06/06/2021	7:55:56	38.059720	-0.943511	0:07:36
456	06/06/2021	7:55:57	38.059657	-0.943423	0:07:37
457	06/06/2021	7:55:58	38.059598	-0.943329	0:07:38

Tabla 5.2: Tabla de datos de localización GPS generada a partir de una vuelta completa a la Universidad de Alicante con GeoVideo.

5.5.2. Diseño del mapa de daños

Obtención del mapa base

El mapa sobre el que se plasmarán las ubicaciones de daños se obtiene a partir de Mapbox, en concreto, generando un token de la acrshortapi de Mapbox desde su página web al crear una cuenta. A partir de esta cuenta, se crea un estilo personalizado en Mapbox Styles (Figura 5.20), modificando la paleta de colores del mapa, las etiquetas de texto del mismo, y las capas

de marcas de objetos destacados para conseguir un mapa sencillo, intuitivo y donde puedan resaltar los daños en la carretera que se superpongan sobre él.

Figura 5.20: Mapa con el estilo personalizado centrado en la Universidad de Alicante.

Marcado de señales de daño sobre el mapa

Para localizar los desperfectos que el modelo encuentre al recorrer una ruta, estos, en primer lugar se guardan en una base de datos que de forma similar a la de la Tabla 5.2, registre el tipo de daño encontrado junto a la fecha, hora, latitud y longitud del daño encontrado. En segundo lugar, los daños se muestran sobre el mapa de la siguiente forma:

- **Mediante un código de color en la ruta recorrida.** Mientras el coche recorre una ruta determinada, dicha ruta se dibuja en el mapa manteniendo en el centro del mismo la posición actual del vehículo y siguiendo el siguiente código de color para pintar la ruta según la gravedad de aquel daño más severo que se ha encontrado en un segundo de vídeo:
  - **Verde.** La ruta se pinta de este color si el modelo no encuentra ningún desperfecto relevante.
  - **Rojo.** Si se encuentran uno o más baches. Es el daño más grave que podemos encontrar en el firme de una carretera.
  - **Naranja.** Si no se han encontrado baches pero sí una o más grietas de cocodrilo. Es un daño de gravedad intermedia ya que tiende a convertirse en un bache con el paso del tiempo.
  - **Amarillo.** Si no se han encontrado baches ni grietas de cocodrilo, pero sí se encuentran una o más grietas laterales o longitudinales. En principio son los daños más leves, aunque pueden convertirse en grietas de cocodrilo y existen algunas grietas con gran profundidad.

Figura 5.21: Captura de una parte de la vuelta a la Universidad de Alicante con la ruta dibujada según el código de color.

- **Mediante señales de aviso flotantes.** Adicionalmente a la ruta con daños por color, se ha diseñado una señal de aviso que representa esquemáticamente cada daño detectable por el modelo (Figura 5.22). Cada vez que en un tramo de la ruta se detecte uno o más desperfectos notables, se mostrará sobre él el símbolo de alerta del daño detectado más grave. Para no saturar demasiado el mapa, se coloca únicamente la señal del daño más grave, y si este daño persiste en un tramo largo y consecutivo, no se vuelve a colocar la misma señal en ese tramo.

Figura 5.22: Diseño de las cuatro señales de aviso de daño.

Velocímetro

Cuando se trata de detectar y clasificar objetos en vídeo, un factor que influye en gran medida es la velocidad de la cámara que graba. Aplicado a este proyecto, esta velocidad repercute en una posible peor estabilización de vídeo, fotogramas borrosos, y una menor tasa de fotogramas por metro de carretera recorrido. Con una tasa de 30 FPS, si el coche que tiene montada la cámara se desplaza a 30 km/h, el vehículo recorrerá aproximadamente 28 cm entre cada par de fotogramas y si se desplaza a 120 km/h la distancia será el cuádruple (1.1 m). Así, a mayor velocidad, el número de fotogramas en los que aparece un mismo objeto

disminuye, resultando en una mayor complejidad para que el modelo lo detecte.

Es por esto que se incluye un velocímetro en la interfaz para comparar el desempeño a distintas velocidades. La velocidad se calcula a partir de los datos GPS mediante la fórmula del semiverseno o *harversine* en inglés (Ecuación 5.1), que devuelve la distancia del camino más corto entre dos puntos de un globo de radio R (el planeta Tierra) conociendo su latitud y longitud y el tiempo que ha transcurrido entre los dos puntos, un segundo.

$$Distancia = 2 * R * arcsin\sqrt{sin^2\left(\frac{\Delta lat}{2}\right) + cos\,(lat1) * cos\,(lat2) * sin^2\left(\frac{\Delta lon}{2}\right)} \quad (5.1)$$

6. Resultados experimentales

En este apartado, se reflejarán los resultados experimentales obtenidos a lo largo del proyecto. Estará centrado en las pruebas más relevantes que se han realizado, para las que se estudiarán sus ventajas y desventajas, se extraerán conclusiones y se compararán entre ellas en base a distintas métricas. Se divide en dos secciones: el entrenamiento del modelo y la aplicación real que usa el mismo.

6.1. Entrenamiento del modelo

Esta sección contendrá todas las pruebas referentes al entrenamiento del modelo predictivo. En primer lugar, se indicarán las especificaciones de hardware de los servidores que se han utilizado para el entrenamiento, y después se pasarán a repasar los resultados obtenidos de los entrenamientos llevados a cabo en función de las técnicas aplicadas.

6.1.1. Especificaciones del hardware utilizado

Todos los experimentos tanto de entrenamiento de modelos como de utilización de los mismos, se han realizado con la GPU por su demostrado mejor desempeño en tareas de *deep learning* frente al uso de la CPU [36]. A continuación, se listan las especificaciones técnicas básicas que definen los dos servidores utilizados durante el desarrollo del presente trabajo. Para cada experimento, se señalará el tipo y número de GPUs usadas, a fin de comparar la velocidad de detección que se puede alcanzar con cada combinación probada.

Servidor Jackson

Es el servidor con el que se comenzó a trabajar.

- **Sistema operativo.** Ubuntu 18.04.5 LTS.
- **CPU.** Intel(R) Core(TM) i9-7920X CPU @ 2.90GHz.
- **RAM.** 64GB 2.133 MHz DDR4.
- **GPUs.** 4 x NVIDIA GeForce RTX 2080 Ti-11GB.

Servidor Lovelace - NVIDIA DGX

Debido a la limitación de 11GB de memoria VRAM de las GPUs RTX 2080 Ti del servidor Jackson, se accedió a este servidor con GPUs con mayor memoria para realizar entrenamientos sin este condicionamiento.

- **Sistema operativo.** Ubuntu 18.04.5 LTS.

- **CPU.** AMD EPYC 7742 64-Core Processor @ 3.30GHz.
- **RAM.** 512GB 2.133 MHz DDR4.
- **GPUs.** 8 x NVIDIA A100-SXM4-40GB.

6.1.2. Experimentos

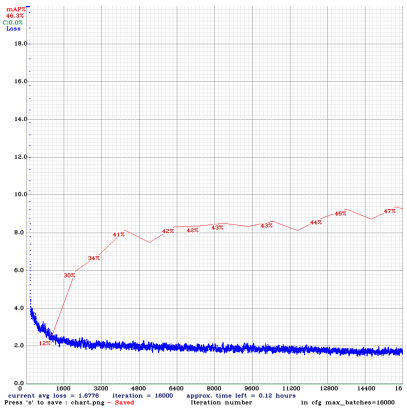
En este apartado se detallarán los entrenamientos más relevantes que se han llevado a cabo. En todos se parte del mismo dataset repartido aleatoriamente en dos conjuntos, uno de entrenamiento con el 80% de las imágenes y otro de test con el restante 20%. Los entrenamientos se compararán entre ellos a partir de las métricas de mAP y F1 que obtengan en el conjunto de test del dataset y la puntuación F1 obtenida en los dos conjuntos de test del reto GRDDC.

Línea base - YOLOv4

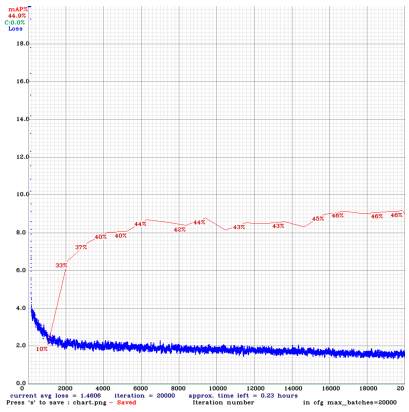
El primer paso es comprobar los resultados que se obtienen con la red de YOLOv4, que al estar más asentada que YOLOv5 (aún en desarrollo) sirve como línea base para comparar con el resto de experimentos. En la Tabla 6.1 se observan los parámetros de entrenamiento y los resultados de las dos redes, y en la Figura 6.1 sus gráficas de entrenamiento.

Modelo	GPU	Tamaño	Batch	Aumentado	mAP	F1	$Test1_{GRDDC}$	$Test2_{GRDDC}$
YOLOv4	RTX 2080Ti	416	16	No	**0.469**	0.493	0.385	0.372
YOLOv4	RTX 2080Ti	608	16	No	0.458	**0.510**	**0.448**	**0.435**

Tabla 6.1: Principales resultados obtenidos con YOLOv4.



(a) Red con imágenes de tamaño 416.



(b) Red con imágenes de tamaño 608.

Figura 6.1: Gráficas de entrenamiento de las dos redes seleccionadas de YOLOv4.

6.1.2.1. Aumento del *dataset*

A continuación, se probarán dos estrategias de aumentado para comprobar cuál de ellas arroja los mejores resultados utilizando el servidor Jackson.

- **YOLOv5 sin aumentar**

  En estos experimentos se parte de la base de YOLOv5 sin ningún tipo de aumento y se entrenan los cuatro modelos de YOLOv5 durante 200 épocas. Los parámetros y resultados de estos entrenamientos se reflejan en la Tabla 6.2. Los resultados son bastante mejorables, siendo el modelo con mayor número de parámetros el que alcanza la mayor puntuación media en los dos tests del reto. También es el modelo con mayor número de parámetros el que tarda más en completar el entrenamiento, con 28 horas en total para las 200 épocas.

Modelo	GPU	Tamaño	Batch	Aumentado	$T_{Época}$ (min)	mAP	F1	$Test1_{GRDDC}$	$Test2_{GRDDC}$
YOLOv5s	RTX 2080Ti	640	32	No	**2.61**	0.229	0.340	0.335	0.339
YOLOv5m	RTX 2080Ti	640	24	No	4.94	0.265	**0.369**	0.425	0.419
YOLOv5l	RTX 2080Ti	512	24	No	5.04	0.257	0.359	0.418	0.412
YOLOv5x	RTX 2080Ti	512	14	No	8.43	**0.267**	0.359	**0.428**	**0.424**

Tabla 6.2: Resultados obtenidos con los cuatro modelos de YOLOv5 sin aumentar.

En la Figura 6.2, se puede observar como en todos los entrenamientos las métricas de precisión del modelo comienzan a decaer y no se recuperan a partir de la época 30. Este comportamiento se debe al *overfitting* producido porque la red recibe repetidamente las mismas imágenes como entrada, adaptándose a estas y dejando así de aprender de forma generalizada.

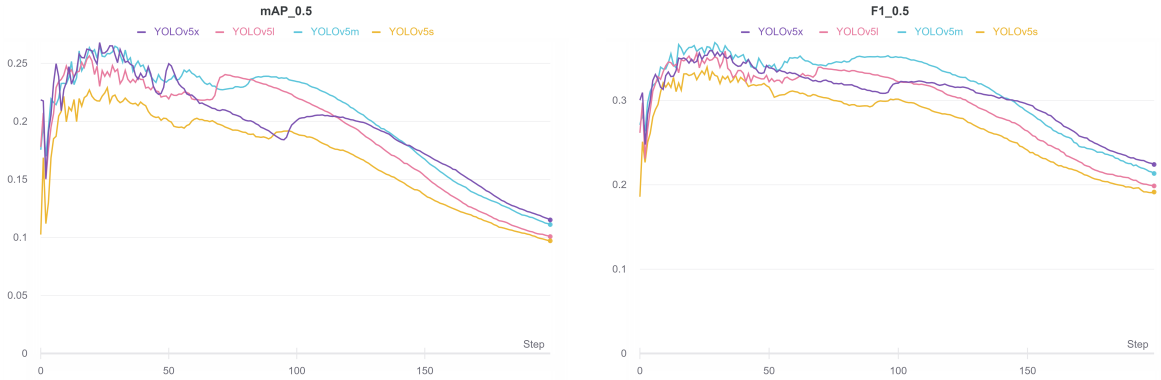


Figura 6.2: Gráficas de mAP y F1 para los modelos de YOLOv5 sin aumentar.

- **Aumentado A - Mosaico**

  Partiendo de los mismos parámetros de entrada que en la versión sin aumentar, se practica un aumentado de tipo mosaico en el que las imágenes con las que se alimenta el entrenamiento se forman con recortes de cuatro imágenes distintas del *dataset*. Se observa una mejoría notable en el rendimiento de todos los modelos, que obtienen unas puntuaciones muy similares (Figura 6.3). En concreto, el modelo YOLOv5x experimenta un incremento de puntuación media entre los tests 1 y 2 del GRDDC de aproximadamente un 33%. Al contrario de lo esperable a priori, el tiempo de entrenamiento no crece al realizar este aumento.

Modelo	GPU	Tamaño	Batch	Aumentado	$T_{Época}$ (min)	mAP	F1	$Test1_{GRDDC}$	$Test2_{GRDDC}$
YOLOv5s	RTX 2080Ti	640	32	A	**2.5**	0.470	0.508	0.569	**0.572**
YOLOv5m	RTX 2080Ti	640	24	A	4.1	**0.479**	**0.519**	0.571	0.564
YOLOv5l	RTX 2080Ti	512	24	A	5.05	0.476	0.515	0.575	0.563
YOLOv5x	RTX 2080Ti	512	14	A	8.33	**0.479**	0.515	**0.576**	0.565

Tabla 6.3: Resultados obtenidos con los cuatro modelos de YOLOv5 con el aumento de mosaico A.

En las gráficas de entrenamiento que se producen con el aumentado de mosaico (Figura 6.3), se puede contemplar una mejor permanencia en el tiempo del aprendizaje del modelo. Si se comparan ambas gráficas, se nota un mayor decaimiento del mAP en comparación con el F1. Probablemente, esta bajada en la métrica mAP se deba a su mayor sensibilidad al *recall* que la métrica F1, que tiene en cuenta a la par la precisión y el *recall*.

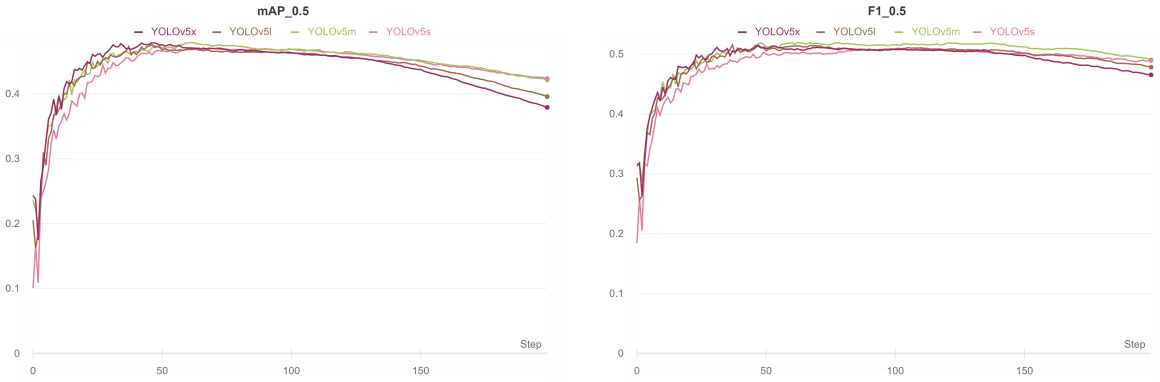


Figura 6.3: Gráficas de mAP y F1 para los modelos de YOLOv5 con el aumento A.

- **Aumentado B - Mosaico, cambio de color, escala y volteo**

  En esta prueba, vista la mejora de rendimiento experimentada al utilizar el aumentado de tipo mosaico, se añaden a este los aumentos por cambio de color, escala y volteo definidos en el apartado anterior de desarrollo. Con este cambio, todas las métricas de calidad son superiores a las del modelo entrenado únicamente con el aumento de tipo mosaico. El modelo con el mayor desempeño de media en los tests del GRDDC es de nuevo el que cuenta con un mayor número de parámetros, YOLOv5x, con un incremento del 7% respecto al aumentado A.

Modelo	GPU	Tamaño	Batch	Aumentado	$T_{Época}$ (min)	mAP	F1	$Test1_{GRDDC}$	$Test2_{GRDDC}$
YOLOv5s	RTX 2080Ti	640	32	B	**2.96**	0.509	0.539	0.588	0.581
YOLOv5m	RTX 2080Ti	640	24	B	5.17	0.527	0.555	**0.616**	0.601
YOLOv5l	RTX 2080Ti	512	24	B	4.4	0.540	**0.561**	0.612	0.599
YOLOv5x	RTX 2080Ti	512	14	B	8.4	**0.542**	0.555	0.613	**0.609**

Tabla 6.4: Resultados obtenidos con los cuatro modelos de YOLOv5 con el aumento B.

Un aspecto que llama la atención sobre los resultados obtenidos, es que el entrenamiento de un modelo con menor número de parámetros (YOLOv5m) dure más que aquel modelo con mayor número de parámetros (YOLOv5l). Esta diferencia en principio carente de sentido, se puede explicar atendiendo a la temperatura de la GPU utilizada para cada entrenamiento (Figura 6.4). Una temperatura alta en una GPU puede afectar a su rendimiento, y esta temperatura puede aumentar por factores externos al entrenamiento.

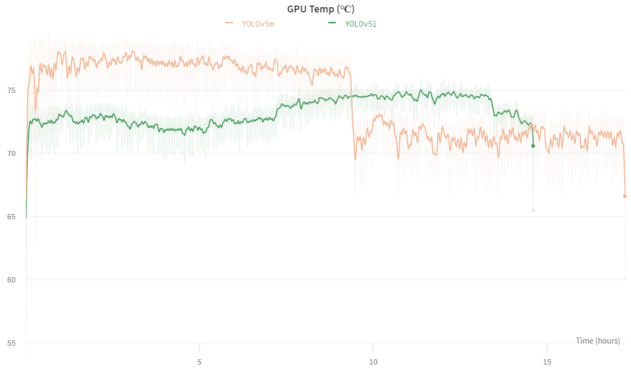


Figura 6.4: Gráfica de temperatura de las GPUs en ºC para los entrenamientos de YOLOv5m y YOLOv5l.

En las gráficas correspondientes (Figura 6.5), se observa como todos los modelos continúan aprendiendo durante gran parte del entrenamiento sin dar lugar al *overfitting* en etapas tempranas. Se atisba un salto de calidad entre los dos modelos con mayor número de parámetros frente a los dos de menor tamaño en la gráfica del mAP.

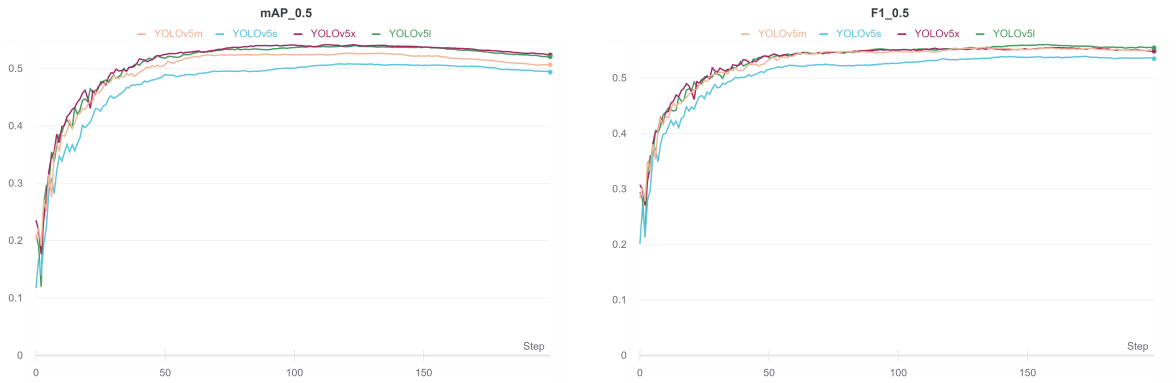


Figura 6.5: Gráficas de mAP y F1 para los modelos de YOLOv5 con el aumento B.

Pero a partir de la etapa 140, las funciones de pérdida en el conjunto de validación comienzan a aumentar. En la Figura 6.6 se muestran las gráficas que produce la función de pérdida de objeto de los cuatro modelos para los conjuntos de entrenamiento y validación. Se observa como el error de detección de objetos en el conjunto de entrenamiento no aumenta, en cambio, en el conjunto de test este error aumenta constantemente a partir de la etapa 140. Esto se debe al ya mencionado *overfitting*, que impide que la red continúe aprendiendo de forma generalizada, por lo que el entrenamiento se detiene.

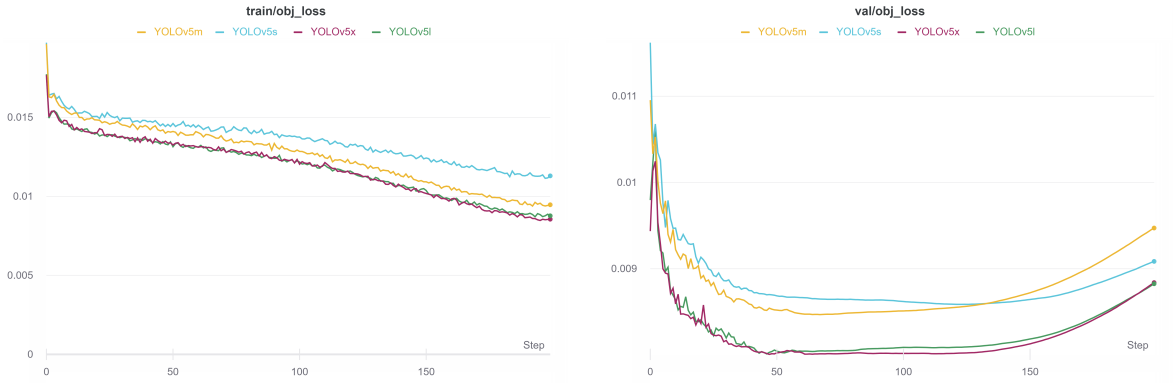


Figura 6.6: Gráficas de la función de pérdida de objetos para el conjunto de entrenamiento (izquierda) y el conjunto de validación (derecha).

6.1.2.2. Incremento del *batch* y el tamaño de imagen

En estos experimentos se utiliza el servidor Lovelace ya que cuenta con GPUs NVIDIA A100 con 40GB de memoria que posibilitan realizar entrenamientos con los modelos más pesados de YOLOv5 haciendo uso de imágenes más grandes y de un mayor *batch* (el número de muestras que se utilizan a la vez en una iteración para entrenar) para hacer que el modelo converja más rápido y mejor.

En la Tabla 6.5, se reflejan los resultados obtenidos en los entrenamientos, en los que se han utilizado imágenes de 640x640 píxeles para alimentar la red en *batches* más grandes, tanto como sea posible dentro de las capacidades de memoria de la GPU para cada red. Estos

resultados, son generalmente mejores que aquellos producidos con un *batch* notablemente menor y un tamaño de imagen mayor en la Tabla 6.4 manteniendo el resto de parámetros de las redes sin modificar. En concreto, el modelo YOLOv5x pasa de una puntuación F1 de 0.555 a 0.565, un aumento de casi un 2%.

Modelo	GPU	Tamaño	Batch	Aumentado	$T_{Época}$ (min)	mAP	F1	$Test1_{GRDDC}$	$Test2_{GRDDC}$
YOLOv5s	A100-SXM4	640	80	B	**1.8**	0.513	0.543	0.602	0.596
YOLOv5m	A100-SXM4	640	72	B	2.47	0.531	0.553	0.620	0.598
YOLOv5l	A100-SXM4	640	64	B	3.03	0.542	0.560	0.619	0.596
YOLOv5x	A100-SXM4	640	50	B	4.84	**0.547**	**0.565**	**0.626**	**0.601**

Tabla 6.5: Resultados obtenidos con los cuatro modelos de YOLOv5 con el aumento B en GPUs A100, con mayor tamaño de imagen y *batch*.

En las gráficas de entrenamiento (Figura 6.7), se puede notar de nuevo cómo aquellos modelos con un mayor número de parámetros obtienen mejores puntuaciones en las métricas mAP y F1. Además, el presupuesto de que con un *batch* mayor se obtiene una generalización más rápida se cumple en este caso, alcanzando el punto de máximo aprendizaje 20 épocas de media antes que con el *batch* reducido.

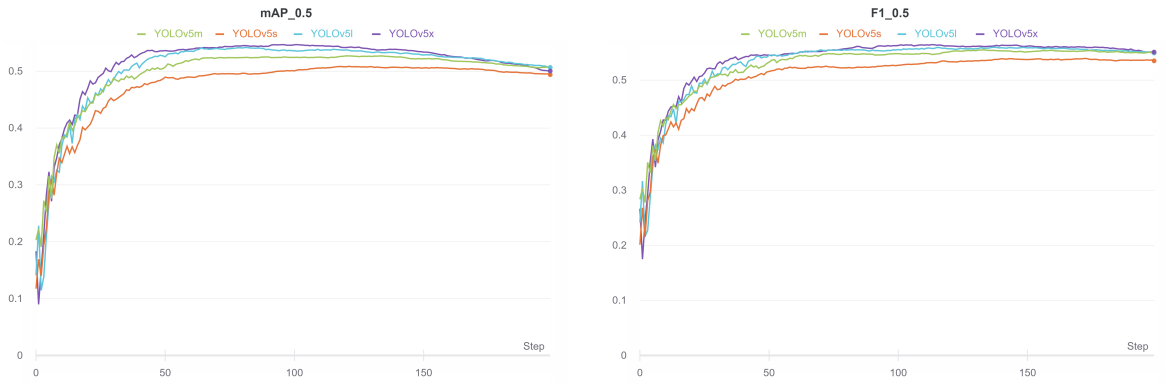


Figura 6.7: Gráficas de mAP y F1 para los modelos de YOLOv5 con el aumento B en GPUs A100, con mayor tamaño de imagen y *batch*.

6.1.2.3. Test Time Augmentation

Con objeto de comprobar el impacto en términos de incremento en la calidad de las métricas y de tiempo de detección que aporta el TTA, se escoge el modelo YOLOv5x entrenado en el apartado anterior por ser el que tiene un tiempo de inferencia más lento y obtener las mejores puntuaciones en las métricas.

En la Tabla 6.6 se observan las pruebas que se han realizado con la GPU RTX 2080Ti, en primer lugar se comprueba el mAP y el tiempo de inferencia del modelo de YOLOv5x sin aplicar TTA, después se hace lo propio aplicando TTA y variando el *batch* de inferencia (el número de imágenes que predice el modelo en una misma iteración). La mejoría en el mAP

del modelo al aplicar TTA es clara, ya que crece un notable 2.5%, pero la detección también resulta mucho más lenta, tardando casi el triple de tiempo en detectar los objetos en una imagen. Pero al aumentar el *batch* de inferencia, se observa como la mejoría en el desempeño del modelo se mantiene (realmente va decayendo pero sin bajar más de un 0.1%) mientras la velocidad de detección aumenta considerablemente superando los 30 FPS necesarios para conseguir una aplicación ejecutable en tiempo real.

Modelo	GPU	Tamaño	Batch	Aumentado	TTA	mAP	T_{Imagen} (ms)	FPS
YOLOv5x	RTX 2080Ti	640	1	B	NO	0.547	**17.2**	**58**
YOLOv5x	RTX 2080Ti	640	1	B	SÍ	**0.561**	47.0	21.3
YOLOv5x	RTX 2080Ti	640	4	B	SÍ	**0.561**	30.3	33
YOLOv5x	RTX 2080Ti	640	8	B	SÍ	**0.561**	27.9	35.8

Tabla 6.6: Resultados de mAP y FPS obtenidos utilizando una GPU RTX 2080Ti aplicando TTA.

En la Tabla 6.7, se plasman los resultados obtenidos al realizar el mismo experimento en una GPU A100-SXM4. El resultado es el mismo, aunque la velocidad de detección aumenta en todos los casos en torno a un 10%.

Modelo	GPU	Tamaño	Batch	Aumentado	TTA	mAP	T_{Imagen} (ms)	FPS
YOLOv5x	A100-SXM4	640	1	B	NO	0.547	**15.4**	**64.9**
YOLOv5x	A100-SXM4	640	1	B	SÍ	**0.561**	42.7	23.4
YOLOv5x	A100-SXM4	640	4	B	SÍ	**0.561**	26.6	37.6
YOLOv5x	A100-SXM4	640	8	B	SÍ	**0.561**	24.8	40.3

Tabla 6.7: Resultados de mAP y FPS obtenidos utilizando una GPU A100-SXM4 aplicando TTA.

6.1.2.4. Model Ensembling

A continuación, se expondrán los resultados obtenidos al aplicar la técnica de ensamblado de modelos. Se realizan dos ensamblados, con 3 y 5 modelos cada uno, formados por aquellos modelos que han alcanzado mejores puntuaciones en las métricas, y utilizando TTA. Se experimenta con ensamblados mayores pero el aumento de precisión es muy pequeño en comparación con el aumento de tiempo de inferencia, por lo que se descarta. Además, se prueban las dos técnicas de ensamblado definidas en el desarrollo, la media y el máximo. Como comparación, se utiliza el modelo de YOLOv5x con TTA del apartado anterior sin ensamblado.

En la Tabla 6.8, se encuentran los resultados para el ensamblado de 3 modelos. La primera conclusión que se extrae es que el ensamblado que toma la predicción con confianza máxima entre todos los modelos funciona mejor que el que los promedia. El ensamblaje de 3 modelos por máximo, obtiene una puntuación un 2% mayor que el mejor modelo sin ensamblaje en mAP. Por contra, la velocidad de inferencia del ensamblaje es mucho menor, de 10 FPS frente a los 33 FPS del modelo sin ensamblaje para un *batch* de inferencia igual a 4. Al aumentar este *batch* la velocidad aumenta sin que el desempeño empeore, pero este aumento de velocidad

es cada vez menor, obteniendo una tasa de 12 FPS con un *batch* de 16.

Ensemble	Tipo	GPU	Tamaño	Batch	TTA	mAP	T_{Imagen} (ms)	FPS	$Test1_{GRDDC}$	$Test2_{GRDDC}$
No	-	RTX 2080Ti	640	4	Sí	0.561	**30.3**	**33**	0.664	0.653
3 Modelos	Media	RTX 2080Ti	640	1	SÍ	0.563	245	4.08	0.665	0.655
3 Modelos	Máximo	RTX 2080Ti	640	1	SÍ	**0.572**	247	4.05	**0.676**	**0.668**
3 Modelos	Media	RTX 2080Ti	640	4	SÍ	0.563	100.9	9.91	0.665	0.655
3 Modelos	Máximo	RTX 2080Ti	640	4	SÍ	**0.572**	100.2	9.98	**0.676**	**0.668**
3 Modelos	Media	RTX 2080Ti	640	8	SÍ	0.563	87.9	11.38	0.665	0.655
3 Modelos	Máximo	RTX 2080Ti	640	8	SÍ	**0.572**	88.4	11.31	**0.676**	**0.668**
3 Modelos	Media	RTX 2080Ti	640	16	SÍ	0.563	82.7	12.09	0.665	0.655
3 Modelos	Máximo	RTX 2080Ti	640	16	SÍ	**0.572**	82.1	12.18	**0.676**	**0.668**

Tabla 6.8: Resultados de velocidad y precisión para el ensamblado de los mejores 3 modelos.

Con el ensamblado de 5 modelos (Tabla 6.9), se logra mejorar el desempeño tanto en mAP como en los dos tests del reto GRDDC en un 0.15% de media. Al mismo tiempo que el desempeño mejora, la velocidad de inferencia empeora, con unos 9 FPS con un *batch* de 16. Además, se confirma que el ensamblado por máximo es más efectivo que el ensamblado por promedio en este caso.

Ensemble	Tipo	GPU	Tamaño	Batch	TTA	mAP	T_{Imagen} (ms)	FPS	$Test1_{GRDDC}$	$Test2_{GRDDC}$
No	-	RTX 2080Ti	640	4	Sí	0.561	**30.3**	**33**	0.664	0.653
5 Modelos	Media	RTX 2080Ti	640	1	SÍ	0.565	344.2	2.91	0.666	0.658
5 Modelos	Máximo	RTX 2080Ti	640	1	SÍ	0.576	332	3.01	**0.677**	**0.670**
5 Modelos	Media	RTX 2080Ti	640	4	SÍ	0.565	143	6.99	0.666	0.658
5 Modelos	Máximo	RTX 2080Ti	640	4	SÍ	0.576	145.1	6.89	**0.677**	**0.670**
5 Modelos	Media	RTX 2080Ti	640	8	SÍ	0.565	122	8.20	0.666	0.658
5 Modelos	Máximo	RTX 2080Ti	640	8	SÍ	0.576	121.1	8.26	**0.677**	**0.670**
5 Modelos	Media	RTX 2080Ti	640	16	SÍ	0.565	107	9.35	0.666	0.658
5 Modelos	Máximo	RTX 2080Ti	640	16	SÍ	0.576	108.5	9.22	**0.677**	**0.670**

Tabla 6.9: Resultados de velocidad y precisión para el ensamblado de los mejores 5 modelos.

En el Anexo A, se pueden ver las métricas que se obtienen del mejor ensamblado de 5 modelos obtenido, con el que se consiguió la mejor puntuación en los dos Tests del GRDDC. Además, se muestra su desempeño en 32 imágenes del conjunto de validación.

6.1.2.5. Clasificación en el ranking de la GRDDC

Los resultados obtenidos con el ensamblado de los 5 mejores modelos del apartado anterior, han resultado ser los que obtienen mejor puntuación F1 en los dos tests del GRDDC 2020. En el Test 1 (Figura 6.8) conseguimos quedar en 4º lugar, con una puntuación de 0.677, y en el Test 2 (Figura 6.9) conseguimos colocarnos en 3er puesto, con una puntuación de 0.670. Estos resultados son tremendamente satisfactorios si los ponemos en contexto, ya que nuestro modelo compite con los modelos de algunos grupos con mucha experiencia en este campo, entre los que destacan:

- **Mì zhì xiǎo AI.** Obtiene la primera posición en ambos tests. Se trata de un grupo de inteligencia arificial de la Universidad de GuanXi, en China.
- **SenseTime.** Considerada por Forbes como la compañía de inteligencia artificial más valiosa del mundo a mediados del pasado año [37], establecida en China.
- **IMSC.** El Integrated Media Systems Center de la Universidad de Southern California, en EEUU.
- **Kattolab.** Un departamento de ciencias de la computación perteneciente a la presigiosa Universidad de Waseda, en Japón
- **CNU-CVLAB.** Un departamento de visión por computador que forma parte de la Universidad Capital Normal, situada en Pekín (China).

Rank	Team name	Score	Upload date
1	秘制小AI	0.732281931730684	2021-05-27 11:55:27 UTC
2	SenseTime	0.721115489416724	2020-12-23 10:08:11 UTC
3	E-LAB	0.690100513470527	2021-04-19 07:22:29 UTC
4	SerenaUA (Ours)	0.677168827737801	2021-07-07 04:01:36 UTC
5	IMSC	0.676180739704447	2020-10-24 10:04:21 UTC
6	GXU507	0.675004908481579	2021-06-08 10:54:11 UTC
7	THE1DOG	0.674996497360093	2021-06-10 15:33:34 UTC
8	光照	0.674980951013792	2021-06-22 05:29:32 UTC
9	芜湖，起飞	0.674949747454426	2021-06-18 14:02:48 UTC
10	sakuchan	0.674945992721765	2021-06-21 00:54:45 UTC

Figura 6.8: Top 10 mejores equipos clasificados en el ranking del Test 1 del GRDDC.

Rank	Team name	Score	Upload date
1	秘制小AI	0.718447751536635	2021-05-27 12:39:32 UTC
2	SenseTime	0.702805783221405	2020-12-23 09:48:46 UTC
3	SerenaUA (Ours)	0.670478469312413	2021-07-06 17:57:23 UTC
4	test	0.666253553218343	2021-05-23 11:17:32 UTC
5	Kattolab	0.666230292231565	2020-12-02 03:12:41 UTC
6	Alone	0.666223104840928	2021-06-20 08:05:10 UTC
7	OverLord	0.666223104840928	2021-06-19 16:23:42 UTC
8	guanbaoguanbao	0.666223104840928	2021-06-01 11:16:25 UTC
9	小辣椒	0.666104292209991	2021-06-13 15:08:44 UTC
10	一栗莎子	0.666021690426726	2021-06-12 08:33:34 UTC

Figura 6.9: Top 10 mejores equipos clasificados en el ranking del Test 2 del GRDDC.

6.2. Aplicación

A fin de probar el funcionamiento de la aplicación desarrollada, en este apartado se aportarán los vídeos generados por la aplicación para una serie de rutas. Además, estas rutas servirán para comprobar el desempeño del modelo bajo distintas condiciones tales como la velocidad del vehículo que capta las imágenes, el tipo de carretera por el que circula, las condiciones lumínicas en el momento de la grabación, o la densidad de daños que tiene el asfalto.

Introducir métricas para juzgar el nivel de rendimiento del sistema sería tramposo pues apreciar la existencia o no de un desperfecto en la carretera y su tipo es un ejercicio subjetivo, y la apreciación se podría modificar para que coincida con la del modelo predictivo. Es por esto que para cada ruta se proporcionará el vídeo en crudo producido por la aplicación, y se destacarán aquellos fallos más relevantes en los que el modelo no funcione como se espera.

Para todos los recorridos se ha utilizado el mismo modelo YOLOv5x con el aumentado B y con la aplicación de la técnica de TTA. Se descarta el uso de ensamblados ya que aunque mejoran el desempeño del modelo, harían muy complicada la detección en tiempo real a 30 FPS sin disponer de servidores muy costosos, algo que va en contra del sistema asequible que se propone.

6.2.1. Ruta 1 - Recorrido en un entorno rural cerca de Orihuela

En este recorrido de 1 km de longitud por una carretera rural cerca de Orihuela, el modelo se comporta generalmente bien, detectando correctamente la mayoría de las grietas longitudinales y de cocodrilo (Figura 6.10) que se encuentran en el vídeo.

Figura 6.10: Detección de una grieta de cocodrilo con una confianza del 78% en la ruta 1.

El mayor fallo que se observa en este recorrido es el FN de lo que a priori debería ser considerado un bache (Figura 6.11). Este fallo puede ocurrir porque no existe ningún bache con unas características parecidas a este en el conjunto de entrenamiento.

Figura 6.11: FN de un bache (recuadro rojo) en la ruta 1.

El vídeo de la ruta 1 completa generado por la aplicación se puede ver en el siguiente enlace.

6.2.2. Ruta 2 - Recorrido alrededor de la Universidad de Alicante

El segundo recorrido consta de una vuelta completa a la carretera que rodea la Universidad de Alicante (UA). En la Figura 6.12, se muestran algunos de los baches que el sistema detecta a lo largo del recorrido.

Figura 6.12: Dos baches detectados en la ruta 2.

La detección de grietas en la ruta se hace de forma recurrente y con bastante acierto, detectándose tramos de la carretera con una grieta longitudinal central que se extiende sobre ellos.

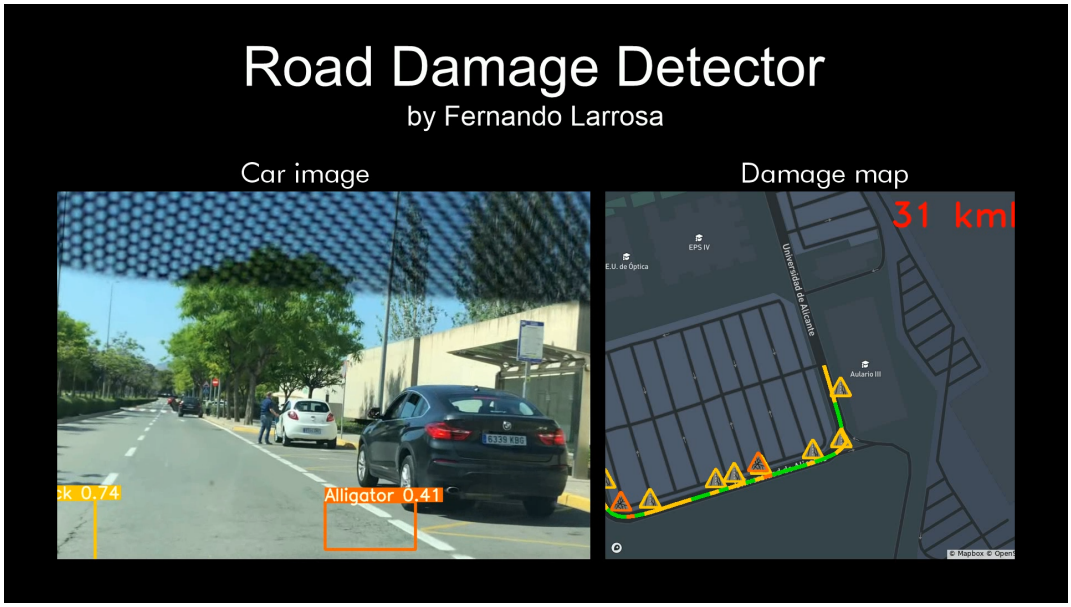


Figura 6.13: Detección de una grieta longitudinal y una grieta de cocodrilo en la ruta 2.

El recorrido alrededor de la UA es bastante largo, por lo que se puede utilizar para encontrar los fallos de detección que más se repiten al utilizar este sistema. Se destacan dos por ser los que se repiten con más frecuencia:

- **Falsos positivos de daños fuera de la carretera.** Se dan cuando el modelo detecta como defecto en la carretera algún objeto fuera de esta. En la Figura 6.14 se observa un ejemplo en el que se detecta una rama como una grieta. Estos errores se podrían solucionar segmentando la carretera como en [38] para detectar fallos sólo en esta, pero se descarta esta técnica por la reducción de velocidad de detección que supone.

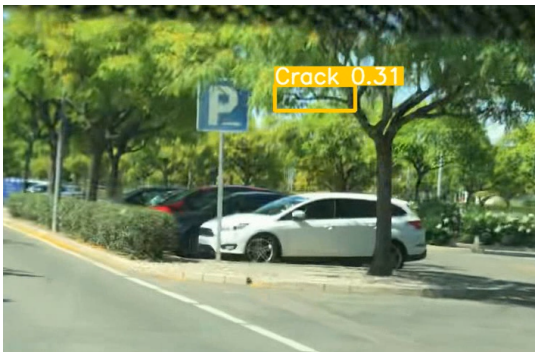


Figura 6.14: FP de una grieta que se confunde con la rama de un árbol en la ruta 1.

- **Falsos positivos de alcantarillas como baches.** Las alcantarillas tienen unas características muy similares a las de los baches, llegando en ocasiones a hundirse y formar un pequeño bache. Por esto, es relativamente sencillo que en imágenes con una calidad baja nuestro modelo falle detectándolas como baches. Normalmente, estos falsos positivos se dan con un porcentaje de confianza no muy alto. Por ejemplo, en la Figura 6.15 se detecta una alcantarilla como bache con una confianza del 31% y en la Figura 6.16 se repite lo mismo con una confianza del 34%. Este mal comportamiento se podía mitigar aumentando el umbral de confianza a partir del cual aceptamos una detección como cierta, incrementando así la precisión del modelo. Pero haciendo esto también se decrementa el *recall* del mismo, ya que verdaderos positivos (TP) como el bache detectado en la esquina inferior derecha de la Figura 6.16 con una confianza del 32% se perderían.

Figura 6.15: FP de una alcantarilla que se confunde con un bache en la ruta 2.

Figura 6.16: FP de una alcantarilla que se confunde con un bache y TP de un bache en la ruta 2.

El vídeo de la ruta 2 completa generado por la aplicación se puede ver en el siguiente enlace.

6.2.3. Ruta 3 - Recorrido por las calles de Elda y Petrer

En la tercera y última ruta, esta vez por las calles de Elda y Petrer, se observa una densidad de daños mucho mayor que en las dos rutas anteriores. En las Figuras 6.17 y 6.18, se muestra un ejemplo de detección de daños de todo tipo; un bache, una grieta longitudinal y tres grietas de cocodrilo. Podemos ver que el modelo tiene un buen comportamiento cuando se enfrenta a imágenes con muchos daños, detectando la mayoría de estos.

Figura 6.17: Detección de una grieta longitudinal y dos grietas de cocodrilo en la ruta 3.

Figura 6.18: Detección de un bache y una grieta de cocodrilo en la ruta 3.

Pero en esta ruta también se observan en alguna ocasión los errores vistos en las anteriores dos rutas. En la Figura 6.19, se ve cómo el modelo detecta un bache que realmente es una alcantarilla con un 39% de confianza, un valor de confianza que comienza a ser alto ya que en el entrenamiento una predicción se considera válida si supera el umbral del 20%.

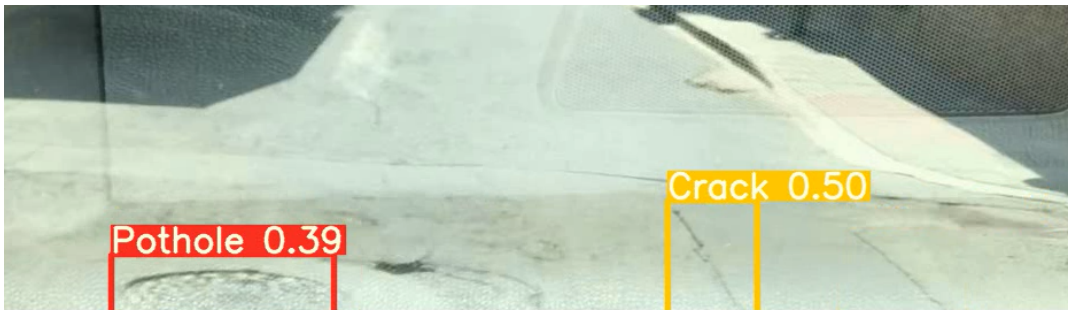


Figura 6.19: FP de una alcantarilla que se confunde con un bache y TP de una grieta longitudinal en la ruta 3.

El vídeo de la ruta 3 completa generado por la aplicación se puede ver en el siguiente enlace.

7. Conclusiones y Trabajos Futuros

Para concluir este TFG, se pasan a listar los objetivos que se han conseguido, y seguidamente se abordarán los trabajos futuros que se proponen. En general, las expectativas creadas al principio del proyecto han sido cumplidas con creces.

- Se ha propuesto un sistema económico capaz de detectar los daños sobre la superficie de las carreteras en tiempo real y con una robustez prometedora aunque mejorable.
- Los datos de un gran *dataset* de *big data* se han manejado y modificado con soltura.
- Se han adquirido unas competencias en el ámbito del *deep learning* que permiten afrontar cualquier reto con unos conocimientos previos.
- Multitud de técnicas se han aplicado que han permitido mejorar notablemente los resultados obtenidos con YOLOv5 en el GRDDC, entre las que se incluyen *Image Augmentation*, *Test Time Augmentation* y *Model Ensembling*.
- Todos las arquitecturas de la red YOLOv5 se han utilizado y comparado entre sí en términos de precisión y velocidad.
- Se ha conseguido un sistema SOTA en la detección de daños en la carretera, avalado por las posiciones en las que se sitúa en nuestro sistema en el ranking de los Test 1 y 2 de la GRDDC, 4º y 3º respectivamente.
- Se ha diseñado una aplicación para detección y mapeado de los daños en las carreteras en tiempo real que utiliza el modelo propuesto.

Con la finalidad de mejorar el sistema propuesto en este TFG, se proponen algunas vías de futuros trabajos:

- Explorar los beneficios de un sistema de pre-procesamiento mediante segmentación que aplicado sobre las imágenes a detectar, envíe al modelo únicamente aquellos píxeles de la imagen que formen parte de la carretera. De esta forma se reduciría en gran medida la detección errónea de daños fuera de la carretera.
- Incluir en el *dataset* una clase dedicada a las alcantarillas. De esta forma, durante el entrenamiento, el modelo podría aprender mejor las diferencias que existen entre un bache y una alcantarilla.
- Ampliar el *dataset* en gran medida con nuevas imágenes, definiendo con detalle cada clase del mismo para conseguir unas anotaciones precisas. Asimismo, unas imágenes con mayor tamaño permitirían al modelo extraer características de mayor calidad y detectar daños más pequeños.

- Desarrollar una aplicación móvil que pueda capturar imágenes de vídeo y datos de localización GPS, y transmitir en tiempo real estos datos mediante redes inalámbricas a un servidor que los pueda procesar y almacenar la información de daños detectados y sus coordenadas en una base de datos.

Bibliografía

[1] A. E. de la Carretera, "Examen a las infraestructuras viarias," 2018.

[2] M. Jahanshahi, F. Jazizadeh, S. Masri, and B. Becerik-Gerber, "Unsupervised approach for autonomous pavement-defect detection and quantification using an inexpensive depth sensor," 2012.

[3] E. Buza, A. Akagic, S. Omanovic, and H. Hasic, "Unsupervised method for detection of high severity distresses on asphalt pavements," in *2017 IEEE 14th International Scientific Conference on Informatics*, pp. 45–50, 2017.

[4] R. Prabhu, "Understanding of convolutional neural network (cnn) — deep learning," 2018.

[5] Q. Chen, X. Gan, W. Huang, J. Feng, and H. Shim, "Road damage detection and classification using mask r-cnn with densenet backbone," *Computers, Materials & Continua*, vol. 65, no. 3, pp. 2201–2215, 2020.

[6] W. H. J. Yin, J. Qu and Q. Chen, "Road damage detection and classification based on multi-level feature pyramids," 2021.

[7] H. Maeda, T. Kashiyama, Y. Sekimoto, T. Seto, and H. Omata, "Generative adversarial network for road damage detection," *Computer-Aided Civil and Infrastructure Engineering*, vol. 36, 06 2020.

[8] R. Xu, H. Lin, K. Lu, L. Cao, and Y. Liu, "A forest fire detection system based on ensemble learning," *Forests*, vol. 12, p. 217, 02 2021.

[9] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: Scalable and efficient object detection," 2020.

[10] StradigiAI, "Image and video understanding: A roadmap for implementation," 2018.

[11] F. Roadware, "Automated pavement and road condition surveys," 2021.

[12] S.-C. Radopoulou and I. Brilakis, "Detection of multiple road defects for pavement condition assessment," 07 2015.

[13] T. F. H. Administration, "Practical guide on quality management procedures for network-level pavement condition data," 2013.

[14] E. Zalama, J. Gómez-García-Bermejo, R. Medina, and J. Llamas, "Road crack detection using visual features extracted by gabor filters," *Computer-Aided Civil and Infrastructure Engineering*, vol. 29, pp. 342–358, 05 2014.

[15] P. de las Naciones Unidas para el Desarrollo, "Informe sobre desarrollo humano," 2020.

[16] W. E. Forum, "The global competitiveness report," 2019.

[17] S. M. Piryonesi, "The application of data analytics to asset management: Deterioration and climate change adaptation in ontario roads," 2019.

[18] NCHRP, "Models for estimating the effects of pavement condition on vehicle operating costs," 2016.

[19] IEEE, "Global road damage detection challenge 2020," 2020.

[20] C. Koch, K. Doycheva, V. Kasi, B. Akinci, and P. Fieguth, "A review on computer vision based defect detection and condition assessment of concrete and asphalt civil infrastructure," *Advanced Engineering Informatics*, vol. 29, 02 2015.

[21] A. C. Yi-Chang Tsai, "Pothole detection and classification using 3d technology and watershed method," 2018.

[22] C. Wu, Z. Wang, S. Hu, J. Lepine, X. Na, D. Ainalis, and M. Stettler, "An automated machine-learning approach for road pothole detection using smartphone sensor data," *Sensors*, vol. 20, no. 19, 2020.

[23] M. L. Rui Fan, "Road damage detection based on unsupervised disparity map segmentation," 2019.

[24] H. Adeli, "Neural networks in civil engineering: 1989–2000," *Computer-Aided Civil and Infrastructure Engineering*, vol. 16, no. 2, pp. 126–142, 2001.

[25] L. Ale, N. Zhang, and L. Li, "Road damage detection using retinanet," in *2018 IEEE International Conference on Big Data (Big Data)*, pp. 5197–5200, 2018.

[26] V. Pham, C. Pham, and T. Dang, "Road damage detection and classification with detectron2 and faster r-cnn," 2020.

[27] W. Wang, B. Wu, S. Yang, and Z. Wang, "Road damage detection and classification with faster r-cnn," in *2018 IEEE International Conference on Big Data (Big Data)*, pp. 5220–5223, 2018.

[28] J. Singh and S. Shekhar, "Road damage detection and classification in smartphone captured images using mask r-cnn," 2018.

[29] K. Doshi and Y. Yilmaz, "Road damage detection using deep ensemble learning," 2020.

[30] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," 2016.

[31] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path aggregation network for instance segmentation," 2018.

[32] D. Arya, H. Maeda, S. K. Ghosh, D. Toshniwal, A. Mraz, T. Kashiyama, and Y. Sekimoto, "Transfer learning-based road damage detection for multiple countries," 2020.

[33] M. Everingham, S. M. A. Eslami, L. V. Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge," 2010.

[34] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," 2020.

[35] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Computer Vision – ECCV 2014* (D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, eds.), (Cham), pp. 740–755, Springer International Publishing, 2014.

[36] E. BUBER and B. DIRI, "Performance analysis and cpu vs gpu comparison for deep learning," in *2018 6th International Conference on Control Engineering Information Technology (CEIT)*, pp. 1–6, 2018.

[37] B. Marr and Forbes, "Meet the world's most valuable ai startup: China's sensetime," 2020.

[38] L. B. Yecheng Lyu and X. Huang, "Road segmentation using cnn and distributed lstm," 2019.

Lista de Acrónimos y Abreviaturas

2D	2 Dimensiones.
3D	3 Dimensiones.
AEC	Agencia Española de la Carretera.
AFP	Adaptive Feature Pooling.
API	Application Programming Interface.
CNN	Redes Neuronales Convolucionales.
COCO	Common Objects in Context.
CPU	Central Processing Unit.
CSPNet	Cross Stage Partial Network.
CSV	Comma-Separated Values.
FLOPS	Floating-point Operations Per Second.
FN	False Negatives.
FP	False Positives.
FPN	Feature Pyramid Network.
FPS	Fotogramas Por Segundo.
GPS	Global Positioning System.
GPU	Graphics Processing Unit.
GRDDC	Global Road Damage Detection Challenge.
HD	High Definition.
HSV	Hue Saturation Value.
IDH	Índice del Desarrollo Humano.
IEEE	Institute of Electrical and Electronics Engineers.
iOS	iPhone Operating System.
IoU	Intersection over Union.
IRI	Índice de Regularidad Internacional.
mAP	Mean Average Precision.
NMS	Non Maxima Supression.
PANet	Path Aggregation Network.
PNG	Portable Network Graphics.
RAM	Random Acess Memory.
RGB	Red Green Blue.
SDKs	Software Development Kits.
SOTA	State Of The Art.
TFG	Trabajo Fin de Grado.
TP	True Positives.
TTA	Test Time Augmentation.

UA	Universidad de Alicante.
VOC	Visual Object Classes.
VRAM	Video Random Access Memory.
XML	Extensible Markup Language.
YOLO	You Only Look Once.

A. Anexo 1 - Métricas del ensamblado de 5 modelos con TTA

A continuación, se aportan las métricas correspondientes al desempeño del ensamblado de 5 modelos con TTA en el conjunto de validación del *dataset*. En la Figura A.1 se observa la matriz de confusión que se produce, obteniendo el mayor *recall* en la clase D20 (grietas de cocodrilo) con un 73% de los objetos de esta clase detectados correctamente. Por contra, la clase con el peor *recall* es la D10, correspondiente a las grietas laterales con sólo un 53% de los objetos de esta clase detectados correctamente.

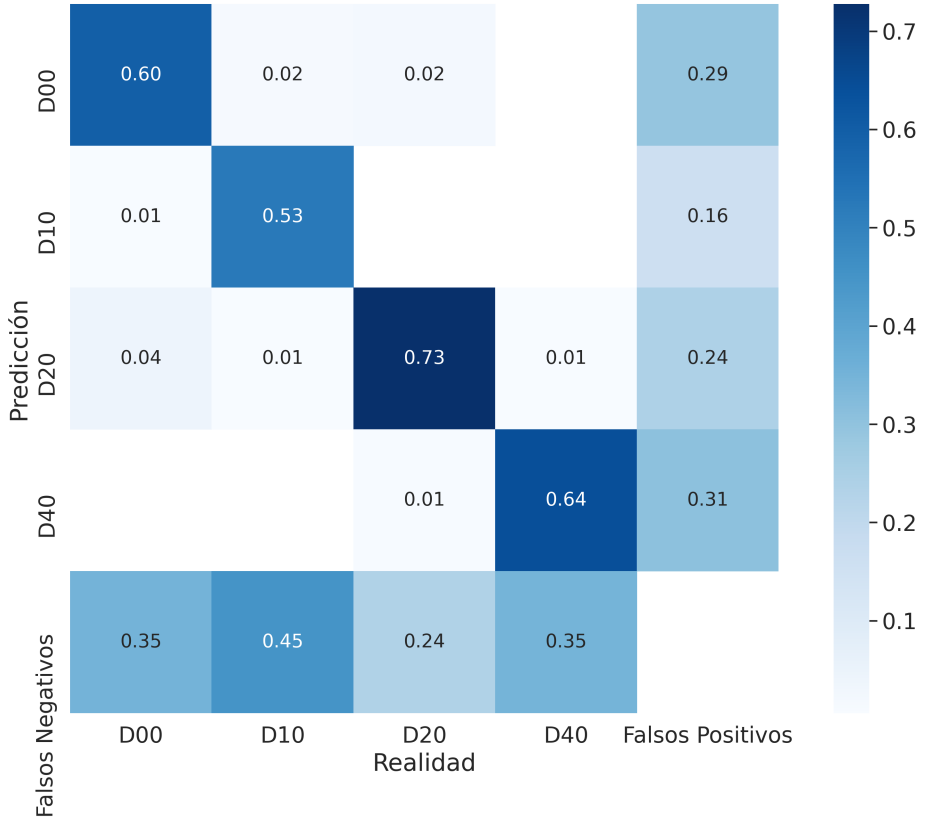
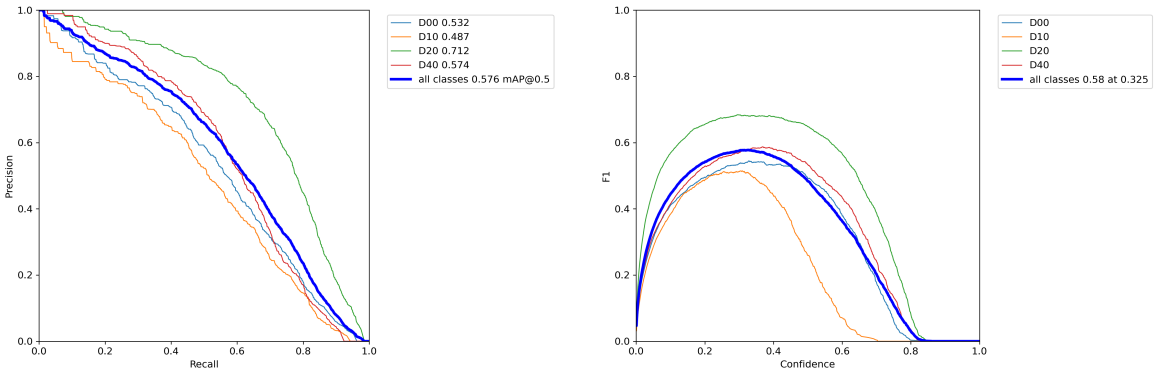


Figura A.1: Matriz de confusión.

En la Figura A.2, se observan las curvas de P-R y F1. La primera demuestra que el mAP o el área bajo la curva es del 57.6% de media entre las cuatro clases, y la segunda muestra que el umbral de confianza que maximiza la puntuación F1 es 0.325. En ambas se vuelve a observar que la clase que mejor detecta el modelo es la D20.



(a) Curva Precision-Recall. **(b)** Curva puntuación F1.

Figura A.2: Curvas de PR y F1.

Finalmente, se exponen las predicciones producidas por el modelo sobre dos grupos de 16 imágenes del conjunto de validación. Estas predicciones, se acompañan de las etiquetas de *ground truth* que el *dataset* contiene de las imágenes. De esta forma, en la Figura A.3 están las imágenes etiquetadas del primer grupo de 16 imágenes, y en la Figura A.4 las detecciones producidas por el modelo y su confianza sobre ellas. De la misma forma, en la Figura A.5 están las imágenes etiquetadas del segundo grupo de 16 imágenes, y en la Figura A.6 las detecciones producidas por el modelo y su confianza sobre ellas.

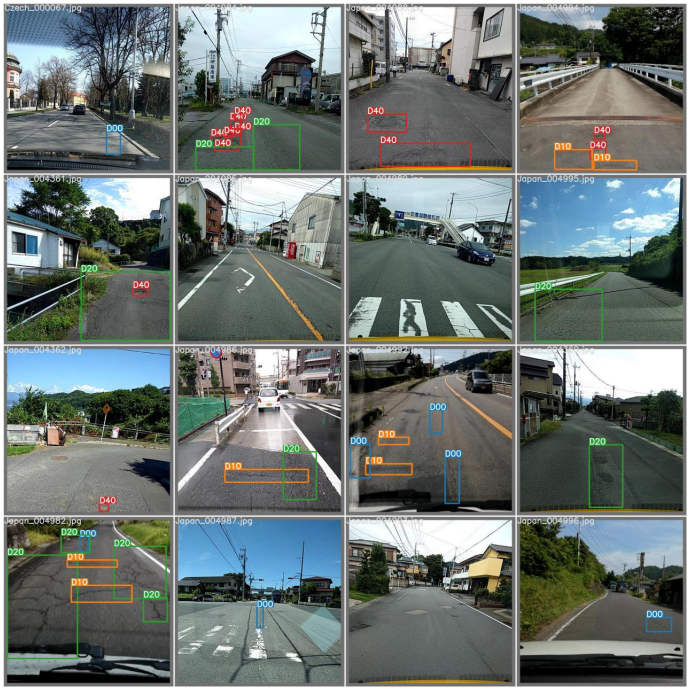


Figura A.3: Etiquetas del primer grupo de 16 imágenes del conjunto de validación.

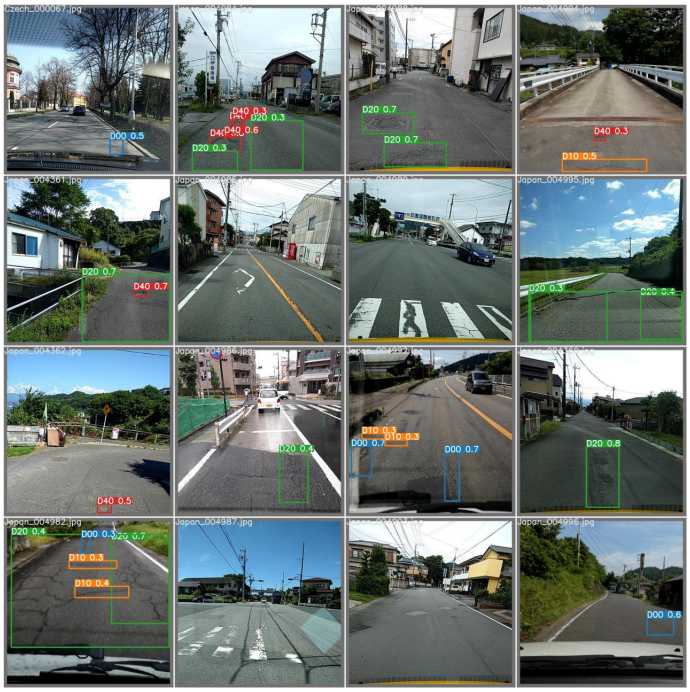


Figura A.4: Predicciones producidas por el modelo para el primer grupo de 16 imágenes del conjunto de validación.

Figura A.5: Etiquetas del segundo grupo de 16 imágenes del conjunto de validación.

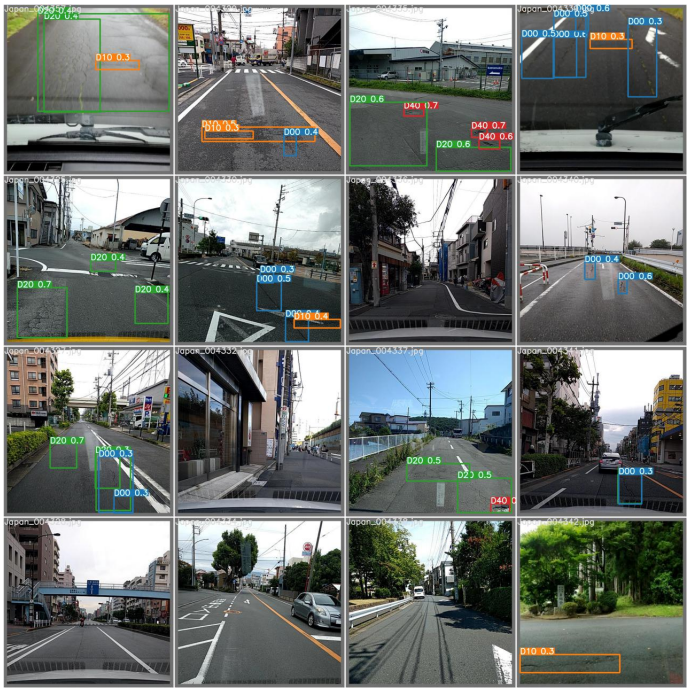


Figura A.6: Predicciones producidas por el modelo para el segundo grupo de 16 imágenes del conjunto de validación.